Cost of Living

September 23, 2021

[12]:
```python
import numpy as np
import pandas as pd

import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px

import folium
from folium import Circle
from geopy import Nominatim

from sklearn.preprocessing import MinMaxScaler
from IPython.display import display
```

[13]:
```python
city = pd.read_csv('cost-of-living.csv')
```

[14]:
```python
city.head()
```

[14]:
```
                                       Unnamed: 0  \
0                       Meal, Inexpensive Restaurant
1  Meal for 2 People, Mid-range Restaurant, Three…
2     McMeal at McDonalds (or Equivalent Combo Meal)
3                  Domestic Beer (0.5 liter draught)
4                   Imported Beer (0.33 liter bottle)

   Saint Petersburg, Russia  Istanbul, Turkey  Izmir, Turkey  \
0                      7.34              4.58           3.06
1                     29.35             15.28          12.22
2                      4.40              3.82           3.06
3                      2.20              3.06           2.29
4                      2.20              3.06           2.75

   Helsinki, Finland  Chisinau, Moldova  Milan, Italy  Cairo, Egypt  \
0              12.00               4.67          15.0          3.38
1              65.00              20.74          60.0         17.48
2               8.00               4.15           8.0          4.51
3               6.50               1.04           5.0          1.69
4               6.75               1.43           5.0          2.82
```

```
   Banja Luka, Bosnia And Herzegovina  Baku, Azerbaijan  …  Lviv, Ukraine  \
0                                3.58              5.27  …           3.75
1                               22.99             23.73  …          18.76
2                                3.58              4.22  …           3.56
3                                1.02              0.84  …           1.50
4                                1.53              2.11  …           1.50

   Novosibirsk, Russia  Bursa, Turkey  Brussels, Belgium  Jerusalem, Israel  \
0                 5.72           3.82               15.0              15.56
1                22.01          11.47               60.0              62.24
2                 3.67           3.06                8.2              12.97
3                 1.10           2.37                4.0               7.26
4                 2.20           3.06                4.0               7.26

   Melbourne, Australia  Perth, Australia  Sydney, Australia  \
0                 10.22             12.43              11.81
1                 49.54             56.55              54.37
2                  7.12              7.32               7.15
3                  5.57              5.90               4.97
4                  5.57              5.59               4.97

   Alexandria, Egypt  Quito, Ecuador
0               2.81            3.59
1              14.06           31.45
2               3.38            5.39
3               1.69            1.35
4               2.81            2.70

[5 rows x 161 columns]
```

[15]:
```
locator = Nominatim(user_agent="myGeocoder")
location = locator.geocode("Saint Petersburg, Russia")
```

[16]:
```
print("Latitude = {}, Longitude = {}".format(location.latitude, location.
 ↪longitude))
```

```
Latitude = 59.917857350000006, Longitude = 30.380619357025516
```

[17]:
```
city = city.T
city.head()
```

[17]:
```
                                                    0  \
Unnamed: 0                 Meal, Inexpensive Restaurant
Saint Petersburg, Russia                           7.34
Istanbul, Turkey                                   4.58
Izmir, Turkey                                      3.06
Helsinki, Finland                                    12
```

```
                                                                  1   \
Unnamed: 0                  Meal for 2 People, Mid-range Restaurant, Three…
Saint Petersburg, Russia                                          29.35
Istanbul, Turkey                                                  15.28
Izmir, Turkey                                                     12.22
Helsinki, Finland                                                    65

                                                               2   \
Unnamed: 0                  McMeal at McDonalds (or Equivalent Combo Meal)
Saint Petersburg, Russia                                       4.4
Istanbul, Turkey                                              3.82
Izmir, Turkey                                                 3.06
Helsinki, Finland                                                8

                                                   3   \
Unnamed: 0                  Domestic Beer (0.5 liter draught)
Saint Petersburg, Russia                           2.2
Istanbul, Turkey                                  3.06
Izmir, Turkey                                     2.29
Helsinki, Finland                                  6.5

                                                   4   \
Unnamed: 0                  Imported Beer (0.33 liter bottle)
Saint Petersburg, Russia                           2.2
Istanbul, Turkey                                  3.06
Izmir, Turkey                                     2.75
Helsinki, Finland                                 6.75

                                                5   \
Unnamed: 0                  Coke/Pepsi (0.33 liter bottle)
Saint Petersburg, Russia                       0.76
Istanbul, Turkey                               0.64
Izmir, Turkey                                  0.61
Helsinki, Finland                              2.66

                                            6   \
Unnamed: 0                  Water (0.33 liter bottle)
Saint Petersburg, Russia                   0.53
Istanbul, Turkey                           0.24
Izmir, Turkey                              0.22
Helsinki, Finland                          1.89

                                           7   \
Unnamed: 0                  Milk (regular), (1 liter)
Saint Petersburg, Russia                  0.98
Istanbul, Turkey                          0.71
```

```
Izmir, Turkey                                0.65
Helsinki, Finland                            0.96

                                                    8  \
Unnamed: 0                 Loaf of Fresh White Bread (500g)
Saint Petersburg, Russia                               0.71
Istanbul, Turkey                                       0.36
Izmir, Turkey                                          0.38
Helsinki, Finland                                      2.27

                                         9  …              45  \
Unnamed: 0                 Eggs (regular) (12)  …  Lettuce (1 head)
Saint Petersburg, Russia                1.18  …              0.86
Istanbul, Turkey                        1.62  …              0.61
Izmir, Turkey                           1.51  …              0.57
Helsinki, Finland                       2.02  …               2.3

                                          46                  47  \
Unnamed: 0                 Cappuccino (regular)  Rice (white), (1kg)
Saint Petersburg, Russia                  1.96                0.92
Istanbul, Turkey                          1.84                 1.3
Izmir, Turkey                             1.56                1.31
Helsinki, Finland                         3.87                2.13

                                    48            49           50  \
Unnamed: 0                 Tomato (1kg)  Banana (1kg)  Onion (1kg)
Saint Petersburg, Russia          1.91          0.89         0.48
Istanbul, Turkey                   0.8          1.91         0.62
Izmir, Turkey                      0.7          1.78         0.58
Helsinki, Finland                 2.91          1.61         1.25

                                                                      51  \
Unnamed: 0                 Beef Round (1kg) (or Equivalent Back Leg Red M…
Saint Petersburg, Russia                                            7.18
Istanbul, Turkey                                                    9.73
Izmir, Turkey                                                       8.61
Helsinki, Finland                                                  12.34

                                                                      52  \
Unnamed: 0                 Toyota Corolla 1.6l 97kW Comfort (Or Equivalen…
Saint Petersburg, Russia                                         19305.3
Istanbul, Turkey                                                 20874.7
Izmir, Turkey                                                    20898.8
Helsinki, Finland                                                24402.8

                                                                      53  \
Unnamed: 0                 Preschool (or Kindergarten), Full Day, Private…
```

```
Saint Petersburg, Russia                                      411.83
Istanbul, Turkey                                              282.94
Izmir, Turkey                                                 212.18
Helsinki, Finland                                              351.6

                                                                  54
Unnamed: 0              International Primary School, Yearly for 1 Child
Saint Petersburg, Russia                                     5388.86
Istanbul, Turkey                                             6905.43
Izmir, Turkey                                                4948.41
Helsinki, Finland                                               1641

[5 rows x 55 columns]
```

[18]:
```python
city.rename(columns=city.iloc[0], inplace = True)
city.drop(city.index[0], inplace = True)
city.head()
```

[18]:
```
                         Meal, Inexpensive Restaurant  \
Saint Petersburg, Russia                         7.34
Istanbul, Turkey                                 4.58
Izmir, Turkey                                    3.06
Helsinki, Finland                                  12
Chisinau, Moldova                                4.67

                         Meal for 2 People, Mid-range Restaurant, Three-course
\
Saint Petersburg, Russia                                              29.35
Istanbul, Turkey                                                      15.28
Izmir, Turkey                                                         12.22
Helsinki, Finland                                                        65
Chisinau, Moldova                                                     20.74

                         McMeal at McDonalds (or Equivalent Combo Meal)  \
Saint Petersburg, Russia                                             4.4
Istanbul, Turkey                                                    3.82
Izmir, Turkey                                                       3.06
Helsinki, Finland                                                      8
Chisinau, Moldova                                                   4.15

                         Domestic Beer (0.5 liter draught)  \
Saint Petersburg, Russia                               2.2
Istanbul, Turkey                                      3.06
Izmir, Turkey                                         2.29
Helsinki, Finland                                      6.5
Chisinau, Moldova                                     1.04
```

	Imported Beer (0.33 liter bottle) \
Saint Petersburg, Russia	2.2
Istanbul, Turkey	3.06
Izmir, Turkey	2.75
Helsinki, Finland	6.75
Chisinau, Moldova	1.43

	Coke/Pepsi (0.33 liter bottle) \
Saint Petersburg, Russia	0.76
Istanbul, Turkey	0.64
Izmir, Turkey	0.61
Helsinki, Finland	2.66
Chisinau, Moldova	0.64

	Water (0.33 liter bottle)	Milk (regular), (1 liter) \
Saint Petersburg, Russia	0.53	0.98
Istanbul, Turkey	0.24	0.71
Izmir, Turkey	0.22	0.65
Helsinki, Finland	1.89	0.96
Chisinau, Moldova	0.44	0.68

	Loaf of Fresh White Bread (500g)	Eggs (regular) (12) \
Saint Petersburg, Russia	0.71	1.18
Istanbul, Turkey	0.36	1.62
Izmir, Turkey	0.38	1.51
Helsinki, Finland	2.27	2.02
Chisinau, Moldova	0.33	1.11

	…	Lettuce (1 head)	Cappuccino (regular) \
Saint Petersburg, Russia	…	0.86	1.96
Istanbul, Turkey	…	0.61	1.84
Izmir, Turkey	…	0.57	1.56
Helsinki, Finland	…	2.3	3.87
Chisinau, Moldova	…	0.84	1.25

	Rice (white), (1kg)	Tomato (1kg)	Banana (1kg) \
Saint Petersburg, Russia	0.92	1.91	0.89
Istanbul, Turkey	1.3	0.8	1.91
Izmir, Turkey	1.31	0.7	1.78
Helsinki, Finland	2.13	2.91	1.61
Chisinau, Moldova	0.93	1.56	1.37

	Onion (1kg) \
Saint Petersburg, Russia	0.48
Istanbul, Turkey	0.62
Izmir, Turkey	0.58
Helsinki, Finland	1.25

```
Chisinau, Moldova              0.59

                            Beef Round (1kg) (or Equivalent Back Leg Red Meat)  \
Saint Petersburg, Russia                                                 7.18
Istanbul, Turkey                                                         9.73
Izmir, Turkey                                                            8.61
Helsinki, Finland                                                       12.34
Chisinau, Moldova                                                        5.37

                            Toyota Corolla 1.6l 97kW Comfort (Or Equivalent New
Car)  \
Saint Petersburg, Russia                                              19305.3
Istanbul, Turkey                                                      20874.7
Izmir, Turkey                                                         20898.8
Helsinki, Finland                                                     24402.8
Chisinau, Moldova                                                     17238.1

                            Preschool (or Kindergarten), Full Day, Private, Monthly
for 1 Child  \
Saint Petersburg, Russia                                               411.83
Istanbul, Turkey                                                       282.94
Izmir, Turkey                                                          212.18
Helsinki, Finland                                                       351.6
Chisinau, Moldova                                                      210.52

                            International Primary School, Yearly for 1 Child
Saint Petersburg, Russia                                     5388.86
Istanbul, Turkey                                             6905.43
Izmir, Turkey                                                4948.41
Helsinki, Finland                                               1641
Chisinau, Moldova                                             2679.3

[5 rows x 55 columns]
```

[19]:
```python
city = city.reset_index()

# lets rename the index column to location
city = city.rename(columns={'index': 'Location'})
city.head()
```

[19]:
```
                   Location Meal, Inexpensive Restaurant  \
0  Saint Petersburg, Russia                          7.34
1          Istanbul, Turkey                          4.58
2             Izmir, Turkey                          3.06
3         Helsinki, Finland                            12
4         Chisinau, Moldova                          4.67
```

```
  Meal for 2 People, Mid-range Restaurant, Three-course  \
0                                             29.35
1                                             15.28
2                                             12.22
3                                                65
4                                             20.74

  McMeal at McDonalds (or Equivalent Combo Meal)  \
0                                            4.4
1                                           3.82
2                                           3.06
3                                              8
4                                           4.15

  Domestic Beer (0.5 liter draught) Imported Beer (0.33 liter bottle)  \
0                               2.2                               2.2
1                              3.06                              3.06
2                              2.29                              2.75
3                               6.5                              6.75
4                              1.04                              1.43

  Coke/Pepsi (0.33 liter bottle) Water (0.33 liter bottle)   \
0                           0.76                      0.53
1                           0.64                      0.24
2                           0.61                      0.22
3                           2.66                      1.89
4                           0.64                      0.44

  Milk (regular), (1 liter) Loaf of Fresh White Bread (500g)  …  \
0                      0.98                             0.71  …
1                      0.71                             0.36  …
2                      0.65                             0.38  …
3                      0.96                             2.27  …
4                      0.68                             0.33  …

  Lettuce (1 head) Cappuccino (regular) Rice (white), (1kg) Tomato (1kg)  \
0             0.86                 1.96                0.92         1.91
1             0.61                 1.84                 1.3          0.8
2             0.57                 1.56                1.31          0.7
3              2.3                 3.87                2.13         2.91
4             0.84                 1.25                0.93         1.56

  Banana (1kg) Onion (1kg) Beef Round (1kg) (or Equivalent Back Leg Red Meat)  \
0         0.89        0.48                                               7.18
1         1.91        0.62                                               9.73
2         1.78        0.58                                               8.61
3         1.61        1.25                                              12.34
```

```
4        1.37       0.59                                               5.37

  Toyota Corolla 1.6l 97kW Comfort (Or Equivalent New Car)  \
0                                            19305.3
1                                            20874.7
2                                            20898.8
3                                            24402.8
4                                            17238.1

  Preschool (or Kindergarten), Full Day, Private, Monthly for 1 Child  \
0                                             411.83
1                                             282.94
2                                             212.18
3                                              351.6
4                                             210.52

  International Primary School, Yearly for 1 Child
0                                          5388.86
1                                          6905.43
2                                          4948.41
3                                             1641
4                                           2679.3

[5 rows x 56 columns]
```

[20]:
```python
# lets check the column names
city.columns
```

[20]:
```
Index(['Location', 'Meal, Inexpensive Restaurant',
       'Meal for 2 People, Mid-range Restaurant, Three-course',
       'McMeal at McDonalds (or Equivalent Combo Meal)',
       'Domestic Beer (0.5 liter draught)',
       'Imported Beer (0.33 liter bottle)', 'Coke/Pepsi (0.33 liter bottle)',
       'Water (0.33 liter bottle) ', 'Milk (regular), (1 liter)',
       'Loaf of Fresh White Bread (500g)', 'Eggs (regular) (12)',
       'Local Cheese (1kg)', 'Water (1.5 liter bottle)',
       'Bottle of Wine (Mid-Range)', 'Domestic Beer (0.5 liter bottle)',
       'Imported Beer (0.33 liter bottle)', 'Cigarettes 20 Pack (Marlboro)',
       'One-way Ticket (Local Transport)',
       'Chicken Breasts (Boneless, Skinless), (1kg)',
       'Monthly Pass (Regular Price)', 'Gasoline (1 liter)', 'Volkswagen Golf',
       'Apartment (1 bedroom) in City Centre',
       'Apartment (1 bedroom) Outside of Centre',
       'Apartment (3 bedrooms) in City Centre',
       'Apartment (3 bedrooms) Outside of Centre',
       'Basic (Electricity, Heating, Cooling, Water, Garbage) for 85m2
Apartment',
```

```
       '1 min. of Prepaid Mobile Tariff Local (No Discounts or Plans)',
       'Internet (60 Mbps or More, Unlimited Data, Cable/ADSL)',
       'Fitness Club, Monthly Fee for 1 Adult',
       'Tennis Court Rent (1 Hour on Weekend)',
       'Cinema, International Release, 1 Seat',
       '1 Pair of Jeans (Levis 501 Or Similar)',
       '1 Summer Dress in a Chain Store (Zara, H&M, …)',
       '1 Pair of Nike Running Shoes (Mid-Range)',
       '1 Pair of Men Leather Business Shoes',
       'Price per Square Meter to Buy Apartment in City Centre',
       'Price per Square Meter to Buy Apartment Outside of Centre',
       'Average Monthly Net Salary (After Tax)',
       'Mortgage Interest Rate in Percentages (%), Yearly, for 20 Years Fixed-
Rate',
       'Taxi Start (Normal Tariff)', 'Taxi 1km (Normal Tariff)',
       'Taxi 1hour Waiting (Normal Tariff)', 'Apples (1kg)', 'Oranges (1kg)',
       'Potato (1kg)', 'Lettuce (1 head)', 'Cappuccino (regular)',
       'Rice (white), (1kg)', 'Tomato (1kg)', 'Banana (1kg)', 'Onion (1kg)',
       'Beef Round (1kg) (or Equivalent Back Leg Red Meat)',
       'Toyota Corolla 1.6l 97kW Comfort (Or Equivalent New Car)',
       'Preschool (or Kindergarten), Full Day, Private, Monthly for 1 Child',
       'International Primary School, Yearly for 1 Child'],
      dtype='object')
```

[21]:
```
import warnings
warnings.filterwarnings('ignore')

# It can be time consuming
from geopy.extra.rate_limiter import RateLimiter

# 1 - conveneint function to delay between geocoding calls
geocode = RateLimiter(locator.geocode, min_delay_seconds=1)

# 2- - create location column
city['location'] = city['Location'].apply(geocode)

# 3 - create longitude, laatitude and altitude from location column (returns␣
↪tuple)
city['point'] = city['location'].apply(lambda loc: tuple(loc.point) if loc else␣
↪None)

# 4 - split point column into latitude, longitude and altitude columns
city[['latitude', 'longitude', 'altitude']] = pd.DataFrame(city['point'].
↪tolist(), index=city.index)

# lets check the head of the data set
city.head()
```

[21]:

```
                    Location Meal, Inexpensive Restaurant  \
0  Saint Petersburg, Russia                          7.34
1          Istanbul, Turkey                          4.58
2             Izmir, Turkey                          3.06
3         Helsinki, Finland                            12
4         Chisinau, Moldova                          4.67

  Meal for 2 People, Mid-range Restaurant, Three-course  \
0                                              29.35
1                                              15.28
2                                              12.22
3                                                 65
4                                              20.74

  McMeal at McDonalds (or Equivalent Combo Meal)  \
0                                             4.4
1                                            3.82
2                                            3.06
3                                               8
4                                            4.15

  Domestic Beer (0.5 liter draught) Imported Beer (0.33 liter bottle)  \
0                               2.2                               2.2
1                              3.06                              3.06
2                              2.29                              2.75
3                               6.5                              6.75
4                              1.04                              1.43

  Coke/Pepsi (0.33 liter bottle) Water (0.33 liter bottle)   \
0                           0.76                       0.53
1                           0.64                       0.24
2                           0.61                       0.22
3                           2.66                       1.89
4                           0.64                       0.44

  Milk (regular), (1 liter) Loaf of Fresh White Bread (500g)  … Onion (1kg)  \
0                      0.98                             0.71  …        0.48
1                      0.71                             0.36  …        0.62
2                      0.65                             0.38  …        0.58
3                      0.96                             2.27  …        1.25
4                      0.68                             0.33  …        0.59

  Beef Round (1kg) (or Equivalent Back Leg Red Meat)  \
0                                               7.18
1                                               9.73
2                                               8.61
3                                              12.34
```

```
4                                              5.37

  Toyota Corolla 1.6l 97kW Comfort (Or Equivalent New Car)  \
0                                            19305.3
1                                            20874.7
2                                            20898.8
3                                            24402.8
4                                            17238.1

  Preschool (or Kindergarten), Full Day, Private, Monthly for 1 Child  \
0                                             411.83
1                                             282.94
2                                             212.18
3                                              351.6
4                                             210.52

  International Primary School, Yearly for 1 Child  \
0                                          5388.86
1                                          6905.43
2                                          4948.41
3                                             1641
4                                           2679.3

                                            location  \
0  (   -     ,     -             …
1  (İstanbul, Fatih, İstanbul, Marmara Bölgesi, 3…
2  (İzmir, Konak, İzmir, Ege Bölgesi, 35180, Türk…
3  (Helsinki, Helsingin seutukunta, Uusimaa, Etel…
4  (Chișinău, Municipiul Chișinău, Moldova, (47.0…

                                         point   latitude  longitude  \
0  (59.917857350000006, 30.380619357025516, 0.0)  59.917857  30.380619
1                (41.0096334, 28.9651646, 0.0)  41.009633  28.965165
2                (38.4147331, 27.1434119, 0.0)  38.414733  27.143412
3                (60.1674881, 24.9427473, 0.0)  60.167488  24.942747
4                (47.0245117, 28.8322923, 0.0)  47.024512  28.832292

  altitude
0      0.0
1      0.0
2      0.0
3      0.0
4      0.0

[5 rows x 61 columns]
```

[22]:
```python
# lets remove some unnecessary columns from the data
city = city.drop(['location','point','altitude'], axis = 1)

# lets check the column names again
city.columns
```

[22]:
```
Index(['Location', 'Meal, Inexpensive Restaurant',
       'Meal for 2 People, Mid-range Restaurant, Three-course',
       'McMeal at McDonalds (or Equivalent Combo Meal)',
       'Domestic Beer (0.5 liter draught)',
       'Imported Beer (0.33 liter bottle)', 'Coke/Pepsi (0.33 liter bottle)',
       'Water (0.33 liter bottle) ', 'Milk (regular), (1 liter)',
       'Loaf of Fresh White Bread (500g)', 'Eggs (regular) (12)',
       'Local Cheese (1kg)', 'Water (1.5 liter bottle)',
       'Bottle of Wine (Mid-Range)', 'Domestic Beer (0.5 liter bottle)',
       'Imported Beer (0.33 liter bottle)', 'Cigarettes 20 Pack (Marlboro)',
       'One-way Ticket (Local Transport)',
       'Chicken Breasts (Boneless, Skinless), (1kg)',
       'Monthly Pass (Regular Price)', 'Gasoline (1 liter)', 'Volkswagen Golf',
       'Apartment (1 bedroom) in City Centre',
       'Apartment (1 bedroom) Outside of Centre',
       'Apartment (3 bedrooms) in City Centre',
       'Apartment (3 bedrooms) Outside of Centre',
       'Basic (Electricity, Heating, Cooling, Water, Garbage) for 85m2
Apartment',
       '1 min. of Prepaid Mobile Tariff Local (No Discounts or Plans)',
       'Internet (60 Mbps or More, Unlimited Data, Cable/ADSL)',
       'Fitness Club, Monthly Fee for 1 Adult',
       'Tennis Court Rent (1 Hour on Weekend)',
       'Cinema, International Release, 1 Seat',
       '1 Pair of Jeans (Levis 501 Or Similar)',
       '1 Summer Dress in a Chain Store (Zara, H&M, …)',
       '1 Pair of Nike Running Shoes (Mid-Range)',
       '1 Pair of Men Leather Business Shoes',
       'Price per Square Meter to Buy Apartment in City Centre',
       'Price per Square Meter to Buy Apartment Outside of Centre',
       'Average Monthly Net Salary (After Tax)',
       'Mortgage Interest Rate in Percentages (%), Yearly, for 20 Years Fixed-
Rate',
       'Taxi Start (Normal Tariff)', 'Taxi 1km (Normal Tariff)',
       'Taxi 1hour Waiting (Normal Tariff)', 'Apples (1kg)', 'Oranges (1kg)',
       'Potato (1kg)', 'Lettuce (1 head)', 'Cappuccino (regular)',
       'Rice (white), (1kg)', 'Tomato (1kg)', 'Banana (1kg)', 'Onion (1kg)',
       'Beef Round (1kg) (or Equivalent Back Leg Red Meat)',
       'Toyota Corolla 1.6l 97kW Comfort (Or Equivalent New Car)',
       'Preschool (or Kindergarten), Full Day, Private, Monthly for 1 Child',
       'International Primary School, Yearly for 1 Child', 'latitude',
```

```
      'longitude'],
     dtype='object')
```

0.1 Aggregating Features

[23]:
```python
def food(city):
    return int(round((city[['Meal, Inexpensive Restaurant',
       'Domestic Beer (0.5 liter draught)',
       'Imported Beer (0.33 liter bottle)', 'Coke/Pepsi (0.33 liter bottle)',
       'Water (0.33 liter bottle) ', 'Milk (regular), (1 liter)',
       'Loaf of Fresh White Bread (500g)', 'Eggs (regular) (12)',
       'Local Cheese (1kg)', 'Water (1.5 liter bottle)',
       'Bottle of Wine (Mid-Range)', 'Domestic Beer (0.5 liter bottle)',
       'Imported Beer (0.33 liter bottle)', 'Cigarettes 20 Pack (Marlboro)',
       'Chicken Breasts (Boneless, Skinless), (1kg)', 'Apples (1kg)', 'Oranges␣
 ↪(1kg)',
       'Potato (1kg)', 'Lettuce (1 head)', 'Cappuccino (regular)',
       'Rice (white), (1kg)', 'Tomato (1kg)', 'Banana (1kg)', 'Onion (1kg)',
       'Beef Round (1kg) (or Equivalent Back Leg Red Meat)',]].mean()).mean()))

def travel(city):
    return int(round((city[['One-way Ticket (Local Transport)',
                            'Monthly Pass (Regular Price)', 'Gasoline (1␣
 ↪liter)',
                           'Taxi Start (Normal Tariff)', 'Taxi 1km (Normal␣
 ↪Tariff)',
                           'Taxi 1hour Waiting (Normal Tariff)',]].mean()).
 ↪mean()))

def living(city):
    return int(round((city[[ 'Volkswagen Golf',
       'Apartment (1 bedroom) in City Centre',
       'Apartment (1 bedroom) Outside of Centre',
       'Apartment (3 bedrooms) in City Centre',
       'Apartment (3 bedrooms) Outside of Centre',
       'Basic (Electricity, Heating, Cooling, Water, Garbage) for 85m2␣
 ↪Apartment',
         'Price per Square Meter to Buy Apartment in City Centre',
       'Price per Square Meter to Buy Apartment Outside of Centre',
        'Toyota Corolla 1.6l 97kW Comfort (Or Equivalent New Car)',]].mean()).
 ↪mean()))

def lifestyle(city):
    return int(round((city[['1 min. of Prepaid Mobile Tariff Local (No␣
 ↪Discounts or Plans)',
       'Internet (60 Mbps or More, Unlimited Data, Cable/ADSL)',
```

```
        'Fitness Club, Monthly Fee for 1 Adult',
        'Tennis Court Rent (1 Hour on Weekend)',
        'Cinema, International Release, 1 Seat',
        '1 Pair of Jeans (Levis 501 Or Similar)',
        '1 Summer Dress in a Chain Store (Zara, H&M, ...)',
        '1 Pair of Nike Running Shoes (Mid-Range)',
        '1 Pair of Men Leather Business Shoes',
        'Meal for 2 People, Mid-range Restaurant, Three-course',
        'McMeal at McDonalds (or Equivalent Combo Meal)',]].mean()).mean()))

def education(city):
    return int(round((city[['Preschool (or Kindergarten), Full Day, Private,␣
 ↪Monthly for 1 Child',
        'International Primary School, Yearly for 1 Child',]].mean()).mean()))

def income(city):
    return int(round((city[['Average Monthly Net Salary (After Tax)',
        'Mortgage Interest Rate in Percentages (%), Yearly, for 20 Years␣
 ↪Fixed-Rate',]].mean()).mean()))
```

[24]:
```
city['Food'] = city.apply(food, axis = 1)
city['Travel'] = city.apply(travel, axis = 1)
city['Living'] = city.apply(living, axis = 1)
city['Lifestyle'] = city.apply(lifestyle, axis = 1)
city['Education'] = city.apply(education, axis = 1)
city['Income'] = city.apply(income, axis = 1)
```

[25]:
```
# lets split the location to fetch the country names
city['Location'].str.split(', ')[0]
```

[25]:
```
['Saint Petersburg', 'Russia']
```

[26]:
```
# lets apply the same function on whole dataset
city['country'] = city['Location'].str.split(', ')

# lets store the second one in the country column
city['Country'] = city['country'].apply(lambda x: x[1])

#lets check the values in the country column
city['Country'].value_counts()
```

[26]:
```
India        11
Canada        8
Poland        6
Romania       5
Australia     5
             ..
```

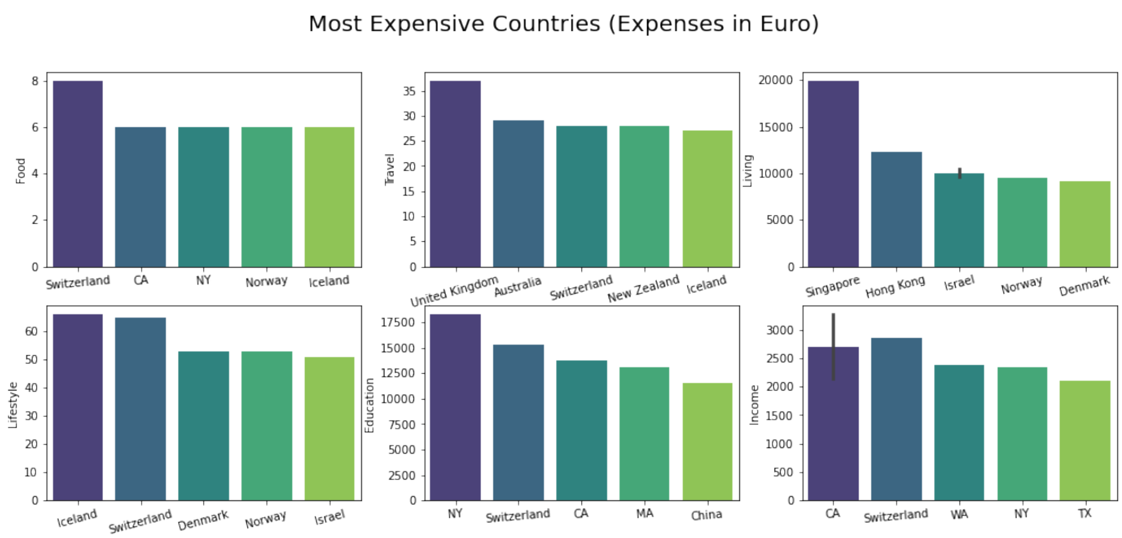
```
Macedonia      1
Taiwan         1
Estonia        1
Kenya          1
Armenia        1
Name: Country, Length: 90, dtype: int64
```

[27]:
```
## lets groupby the Countries with Lifestyle Factors

city[['Country','Food','Travel',
      'Living','Lifestyle','Education','Income']].groupby(['Country']).
 ↪agg('mean').style.background_gradient(cmap = 'Wistia')
```

[27]: <pandas.io.formats.style.Styler at 0x2fafe68>

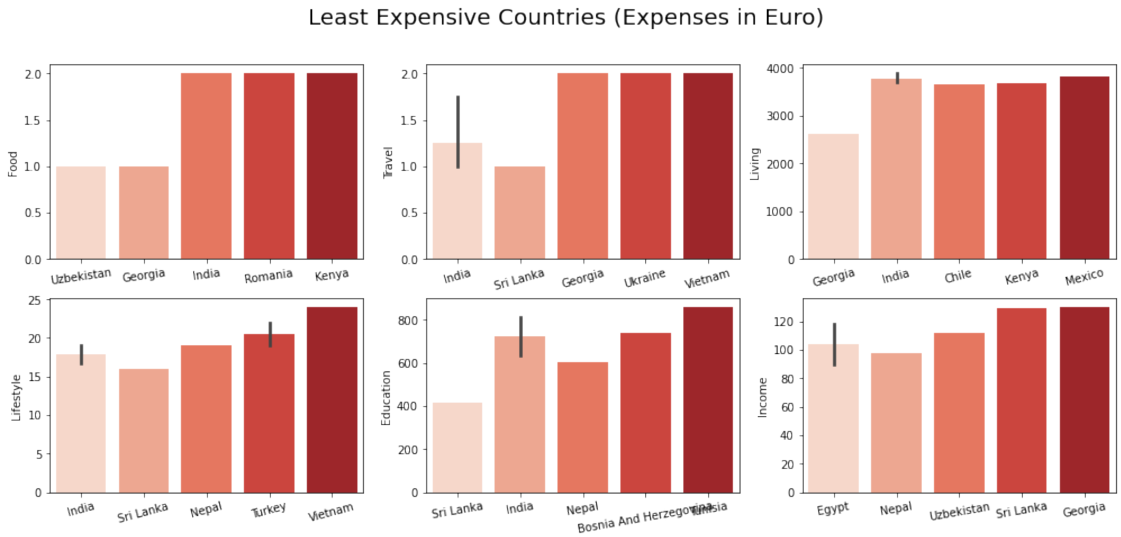
[28]:
```
# Let's check out the Top 5 Most Expensive Countries for Food
plt.rcParams['figure.figsize'] = (17, 7)

plt.subplot(2, 3, 1)
x = city[['Country','Food']].sort_values(by = 'Food', ascending = False).head(5)
sns.barplot(x['Country'], x['Food'], palette = 'viridis')
plt.xticks(rotation = 5)
plt.xlabel(' ')

plt.subplot(2, 3, 2)
x = city[['Country','Travel']].sort_values(by = 'Travel', ascending = False).
 ↪head(5)
sns.barplot(x['Country'], x['Travel'], palette = 'viridis')
plt.xticks(rotation = 16)
plt.xlabel(' ')

plt.subplot(2, 3, 3)
x = city[['Country','Living']].sort_values(by = 'Living', ascending = False).
 ↪head(6)
sns.barplot(x['Country'], x['Living'], palette = 'viridis')
plt.xticks(rotation = 15)
plt.xlabel(' ')

plt.subplot(2, 3, 4)
x = city[['Country','Lifestyle']].sort_values(by = 'Lifestyle', ascending =␣
 ↪False).head(5)
sns.barplot(x['Country'], x['Lifestyle'], palette = 'viridis')
plt.xticks(rotation = 15)
plt.xlabel(' ')

plt.subplot(2, 3, 5)
```

```python
x = city[['Country','Education']].sort_values(by = 'Education', ascending = False).head(5)
sns.barplot(x['Country'], x['Education'], palette = 'viridis')
plt.xticks(rotation = 5)
plt.xlabel(' ')

plt.subplot(2, 3, 6)
x = city[['Country','Income']].sort_values(by = 'Income', ascending = False).head(6)
sns.barplot(x['Country'], x['Income'], palette = 'viridis')
plt.xticks(rotation = 5)
plt.xlabel(' ')

plt.suptitle('Most Expensive Countries (Expenses in Euro)', fontsize = 20)
plt.show()
```



[29]:
```python
# Let's check out the Top 5 Least Expensive Countries
plt.rcParams['figure.figsize'] = (17, 7)

plt.subplot(2, 3, 1)
x = city[['Country','Food']].sort_values(by = 'Food', ascending = True).head(5)
sns.barplot(x['Country'], x['Food'], palette = 'Reds')
plt.xticks(rotation = 8)
plt.xlabel(' ')

plt.subplot(2, 3, 2)
x = city[['Country','Travel']].sort_values(by = 'Travel', ascending = True).head(9)
```

```python
sns.barplot(x['Country'], x['Travel'], palette = 'Reds')
plt.xticks(rotation = 16)
plt.xlabel(' ')

plt.subplot(2, 3, 3)
x = city[['Country','Living']].sort_values(by = 'Living', ascending = True).head(9)
sns.barplot(x['Country'], x['Living'], palette = 'Reds')
plt.xticks(rotation = 15)
plt.xlabel(' ')

plt.subplot(2, 3, 4)
x = city[['Country','Lifestyle']].sort_values(by = 'Lifestyle', ascending = True).head(18)
sns.barplot(x['Country'], x['Lifestyle'], palette = 'Reds')
plt.xticks(rotation = 15)
plt.xlabel(' ')

plt.subplot(2, 3, 5)
x = city[['Country','Education']].sort_values(by = 'Education', ascending = True).head(9)
sns.barplot(x['Country'], x['Education'], palette = 'Reds')
plt.xticks(rotation = 10)
plt.xlabel(' ')

plt.subplot(2, 3, 6)
x = city[['Country','Income']].sort_values(by = 'Income', ascending = True).head(6)
sns.barplot(x['Country'], x['Income'], palette = 'Reds')
plt.xticks(rotation = 10)
plt.xlabel(' ')

plt.suptitle('Least Expensive Countries (Expenses in Euro)', fontsize = 20)
plt.show()
```



[30]:
```
# To find some interesting columns to plot I've sorted them by range.
# Perhaps a better way to do this in future would be by variance.
top_range = (city.describe().loc['min',:]/city.describe().loc['max',:]).
↪sort_values().index[2:22]
list(top_range)
```

[30]: ['Education', 'Travel', 'Income', 'Food', 'Living', 'Lifestyle']

[31]:
```
def color_producer(val):
    if val <= city[item].quantile(.25):
        return 'forestgreen'
    elif val <= city[item].quantile(.50):
        return 'goldenrod'
    elif val <= city[item].quantile(.75):
        return 'darkred'
    else:
        return 'red'
```

[32]:
```
map = folium.Map(location=[city['latitude'].mean(),
                           city['longitude'].mean()],
                 tiles='Stamen Terrain',
                 zoom_start=2)

item = top_range[0]

# Add a bubble map to the base map
for i in range(0,len(city)):
    Circle(
```

```
        location=[city.iloc[i]['latitude'], city.iloc[i]['longitude']],
        radius=120000,
        color=color_producer(city.iloc[i][item])).add_to(map)

print ('Price of: ', item)
map
```

```
Price of:  Education
```

[32]: <folium.folium.Map at 0xa505418>

[33]:
```
map = folium.Map(location=[city['latitude'].mean(),
                           city['longitude'].mean()],
                 tiles='CartoDB dark_matter',
                 zoom_start=2)

item = top_range[1]

# Add a bubble map to the base map
for i in range(0,len(city)):
    Circle(
        location=[city.iloc[i]['latitude'], city.iloc[i]['longitude']],
        radius=120000,
        color=color_producer(city.iloc[i][item])).add_to(map)

print ('Price of: ', item)
map
```

```
Price of:  Travel
```

[33]: <folium.folium.Map at 0xa5056d0>

[34]:
```
map = folium.Map(location=[city['latitude'].mean(),
                           city['longitude'].mean()],
                 tiles='Stamen Toner',
                 zoom_start=2)

item = top_range[2]

# Add a bubble map to the base map
for i in range(0,len(city)):
    Circle(
        location=[city.iloc[i]['latitude'], city.iloc[i]['longitude']],
        radius=120000,
        color=color_producer(city.iloc[i][item])).add_to(map)

print ('Price of: ', item)
map
```

```
Price of:  Income
```

[34]: <folium.folium.Map at 0xb30ad00>

[35]:
```python
map = folium.Map(location=[city['latitude'].mean(),
                           city['longitude'].mean()],
                 tiles='Stamen Watercolor',
                 zoom_start=2)

item = top_range[3]

# Add a bubble map to the base map
for i in range(0,len(city)):
    Circle(
        location=[city.iloc[i]['latitude'], city.iloc[i]['longitude']],
        radius=120000,
        color=color_producer(city.iloc[i][item])).add_to(map)

print ('Price of: ', item)
map
```

```
Price of:  Food
```

[35]: <folium.folium.Map at 0xb385f58>

[36]:
```python
map = folium.Map(location=[city['latitude'].mean(),
                           city['longitude'].mean()],
                 tiles='Open Street Map',
                 zoom_start=2)

item = top_range[4]

# Add a bubble map to the base map
for i in range(0,len(city)):
    Circle(
        location=[city.iloc[i]['latitude'], city.iloc[i]['longitude']],
        radius=120000,
        color=color_producer(city.iloc[i][item])).add_to(map)

print ('Price of: ', item)
map
```

```
Price of:  Living
```

[36]: <folium.folium.Map at 0xb42b178>

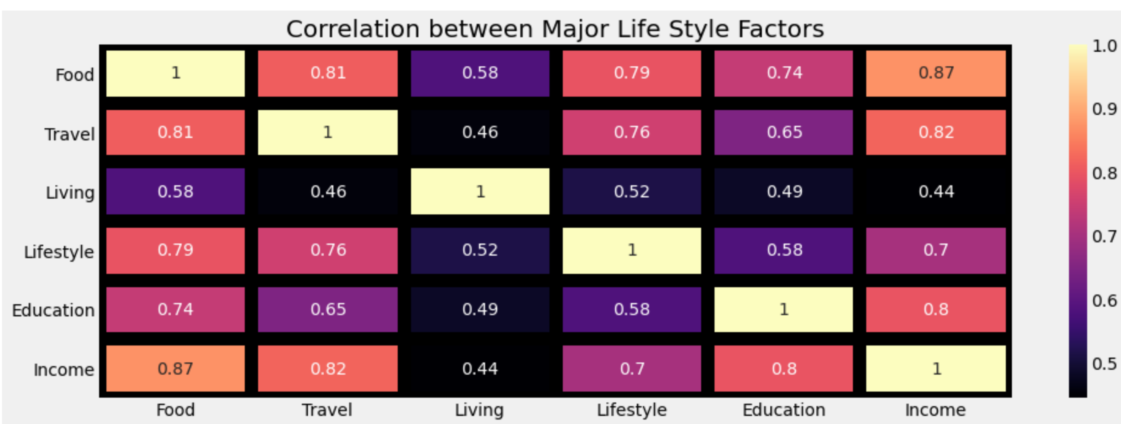
[37]:
```python
map = folium.Map(location=[city['latitude'].mean(),
                           city['longitude'].mean()],
                 tiles='CartoDB Positron',
```

```
                 zoom_start=2)

item = top_range[5]

# Add a bubble map to the base map
for i in range(0,len(city)):
    Circle(
        location=[city.iloc[i]['latitude'], city.iloc[i]['longitude']],
        radius=120000,
        color=color_producer(city.iloc[i][item])).add_to(map)

print ('Price of: ', item)
map
```

```
Price of:  Lifestyle
```

[37]: <folium.folium.Map at 0xb3acf10>

[38]:
```
plt.rcParams['figure.figsize'] = (15, 5)
plt.style.use('fivethirtyeight')

sns.heatmap(city[['Food','Travel','Living','Lifestyle','Education','Income']].
 ↪corr(),
            cmap = 'magma',
            annot = True, linecolor='black', linewidths = 10)
plt.title('Correlation between Major Life Style Factors', fontsize = 20)
plt.show()
```

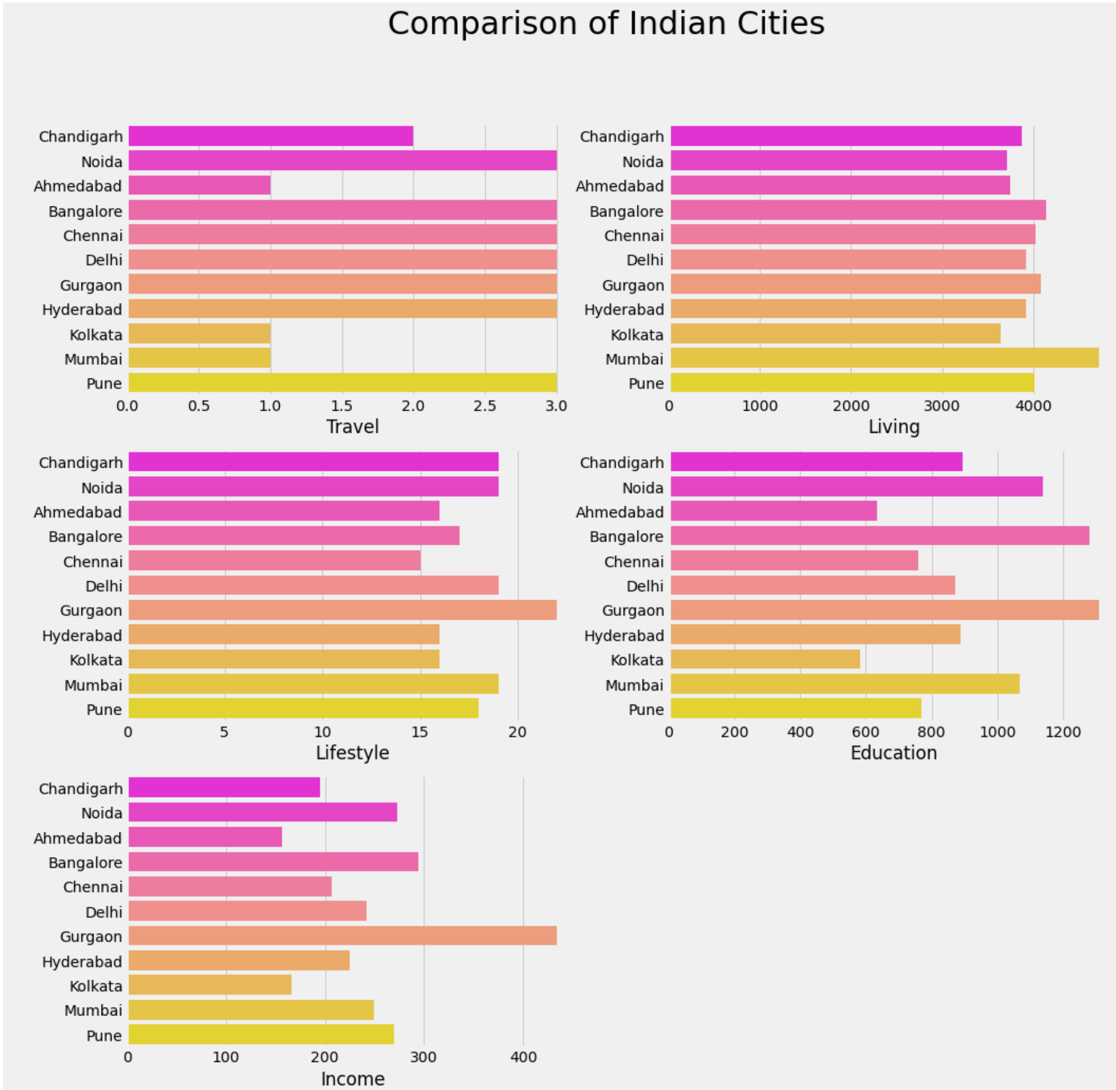


0.2 Comparing Some of the Most Popular Countries in the World

[39]:
```
city[(city['Country'] == 'United Kingdom') | (city['Country'] == 'Australia') |
     (city['Country'] == 'Germany') | (city['Country'] == 'China') |
    (city['Country'] == 'Russia')][['Location','Food',
                'Travel','Living','Lifestyle','Education',
                    'Income']].set_index('Location').sort_values(by = 'Income',
                                ascending = False).style.
 ↪background_gradient(cmap = 'copper')
```

[39]: <pandas.io.formats.style.Styler at 0x9c56e98>

[40]:
```
# let's plot the Indian Cities to understand them better
city['City'] = city['Location'].str.split(', ')
city['City'] = city['City'].apply(lambda x: x[0])

x = city[city['Country'] == 'India']

plt.rcParams['figure.figsize'] = (15, 15)
plt.subplot(3, 2, 1)
sns.barplot(y = x['City'], x = x['Travel'], palette = 'spring')
plt.ylabel(" ")

plt.subplot(3, 2, 2)
sns.barplot(y = x['City'], x = x['Living'], palette = 'spring')
plt.ylabel(" ")

plt.subplot(3, 2, 3)
sns.barplot(y = x['City'], x = x['Lifestyle'], palette = 'spring')
plt.ylabel(" ")

plt.subplot(3, 2, 4)
sns.barplot(y = x['City'], x = x['Education'], palette = 'spring')
plt.ylabel(" ")

plt.subplot(3, 2, 5)
sns.barplot(y = x['City'], x = x['Income'], palette = 'spring')
plt.ylabel(" ")

plt.suptitle('Comparison of Indian Cities', fontsize = 30)
plt.show()
```




```
[41]: # lets find out the List of Most Expensive Countries to Live in
x = city[['Food','Travel','Living','Lifestyle','Education','Income']]
mm = MinMaxScaler()
data = mm.fit_transform(x)
data = pd.DataFrame(data)
data.columns = x.columns
data.head()
```

[41]:

	Food	Travel	Living	Lifestyle	Education	Income
0	0.142857	0.194444	0.139139	0.372549	0.139049	0.074890
1	0.142857	0.138889	0.161588	0.156863	0.177913	0.043424
2	0.142857	0.111111	0.155125	0.078431	0.121129	0.036816
3	0.428571	0.527778	0.252539	0.568627	0.032424	0.346759

```
4  0.142857  0.027778  0.082583   0.333333   0.057568  0.018250
```

[42]:
```python
data['Total Score'] = (data['Food'] + data['Travel'] + data['Living'] +
                       data['Lifestyle'] + data['Education'] + data['Income'])/6

# concat city
cities = city[['City', 'Country']]
data = pd.concat([data, cities], axis = 1)
# lets sort the values
print("Most Expensive Places in the World\n")
data[['Country','City','Total Score']].sort_values(by = 'Total Score',␣
 ↪ascending = False).head(10)
```

```
Most Expensive Places in the World
```

[42]:

	Country	City	Total Score
144	Switzerland	Zurich	0.790375
78	NY	New York	0.670422
100	CA	San Francisco	0.660000
39	Singapore	Singapore	0.576437
70	United Kingdom	London	0.558841
128	Iceland	Reykjavik	0.557198
102	WA	Seattle	0.539670
50	Norway	Oslo	0.534259
47	MA	Boston	0.525962
71	CA	Los Angeles	0.487471

[43]:
```python
# Cheapest places to live

print("Cheapest Places in the World\n")
data[['Country','City','Total Score']].sort_values(by = 'Total Score',␣
 ↪ascending = True).head(10)
```

```
Cheapest Places in the World
```

[43]:

	Country	City	Total Score
123	India	Kolkata	0.042501
106	India	Ahmedabad	0.043629
112	India	Chennai	0.056045
63	Georgia	Tbilisi	0.057976
118	India	Hyderabad	0.060394
81	India	Chandigarh	0.063713
130	India	Pune	0.069141
82	Sri Lanka	Colombo	0.069626
114	India	Delhi	0.070959
127	India	Mumbai	0.071562

0.2.1 Analyzing Cost of Essential Items

[44]:
```
city.columns
```

[44]:
```
Index(['Location', 'Meal, Inexpensive Restaurant',
       'Meal for 2 People, Mid-range Restaurant, Three-course',
       'McMeal at McDonalds (or Equivalent Combo Meal)',
       'Domestic Beer (0.5 liter draught)',
       'Imported Beer (0.33 liter bottle)', 'Coke/Pepsi (0.33 liter bottle)',
       'Water (0.33 liter bottle) ', 'Milk (regular), (1 liter)',
       'Loaf of Fresh White Bread (500g)', 'Eggs (regular) (12)',
       'Local Cheese (1kg)', 'Water (1.5 liter bottle)',
       'Bottle of Wine (Mid-Range)', 'Domestic Beer (0.5 liter bottle)',
       'Imported Beer (0.33 liter bottle)', 'Cigarettes 20 Pack (Marlboro)',
       'One-way Ticket (Local Transport)',
       'Chicken Breasts (Boneless, Skinless), (1kg)',
       'Monthly Pass (Regular Price)', 'Gasoline (1 liter)', 'Volkswagen Golf',
       'Apartment (1 bedroom) in City Centre',
       'Apartment (1 bedroom) Outside of Centre',
       'Apartment (3 bedrooms) in City Centre',
       'Apartment (3 bedrooms) Outside of Centre',
       'Basic (Electricity, Heating, Cooling, Water, Garbage) for 85m2
Apartment',
       '1 min. of Prepaid Mobile Tariff Local (No Discounts or Plans)',
       'Internet (60 Mbps or More, Unlimited Data, Cable/ADSL)',
       'Fitness Club, Monthly Fee for 1 Adult',
       'Tennis Court Rent (1 Hour on Weekend)',
       'Cinema, International Release, 1 Seat',
       '1 Pair of Jeans (Levis 501 Or Similar)',
       '1 Summer Dress in a Chain Store (Zara, H&M, …)',
       '1 Pair of Nike Running Shoes (Mid-Range)',
       '1 Pair of Men Leather Business Shoes',
       'Price per Square Meter to Buy Apartment in City Centre',
       'Price per Square Meter to Buy Apartment Outside of Centre',
       'Average Monthly Net Salary (After Tax)',
       'Mortgage Interest Rate in Percentages (%), Yearly, for 20 Years Fixed-
Rate',
       'Taxi Start (Normal Tariff)', 'Taxi 1km (Normal Tariff)',
       'Taxi 1hour Waiting (Normal Tariff)', 'Apples (1kg)', 'Oranges (1kg)',
       'Potato (1kg)', 'Lettuce (1 head)', 'Cappuccino (regular)',
       'Rice (white), (1kg)', 'Tomato (1kg)', 'Banana (1kg)', 'Onion (1kg)',
       'Beef Round (1kg) (or Equivalent Back Leg Red Meat)',
       'Toyota Corolla 1.6l 97kW Comfort (Or Equivalent New Car)',
       'Preschool (or Kindergarten), Full Day, Private, Monthly for 1 Child',
       'International Primary School, Yearly for 1 Child', 'latitude',
       'longitude', 'Food', 'Travel', 'Living', 'Lifestyle', 'Education',
       'Income', 'country', 'Country', 'City'],
```

```
      dtype='object')
```

[45]:
```
# We know that the Most common things in day to day life are
# Internet, Basic Food ItemS such as Eggs, Milk, Breads, Electricity and Water,␣
↪Taxi Travel

x = city[['Country','City','Milk (regular), (1 liter)',
         'Eggs (regular) (12)', 'Loaf of Fresh White Bread (500g)',
         'Internet (60 Mbps or More, Unlimited Data, Cable/ADSL)',
         'Taxi 1km (Normal Tariff)',
         'Basic (Electricity, Heating, Cooling, Water, Garbage) for 85m2␣
↪Apartment',
          ]]

# lets rename these columns
x = x.rename(columns = {'Milk (regular), (1 liter)':'Milk','Eggs (regular)␣
↪(12)':'Eggs',
                       'Loaf of Fresh White Bread (500g)': 'Bread',
                       'Internet (60 Mbps or More, Unlimited Data, Cable/ADSL)':
↪ 'Internet',
                       'Taxi 1km (Normal Tariff)':'Taxi Travel',
                       'Basic (Electricity, Heating, Cooling, Water, Garbage)␣
↪for 85m2 Apartment':'Electricity and Water'})
x.head()
```

[45]:
```
   Country              City  Milk  Eggs Bread Internet Taxi Travel  \
0   Russia  Saint Petersburg  0.98  1.18  0.71     6.96        0.26
1   Turkey          Istanbul  0.71  1.62  0.36     14.2        0.47
2   Turkey             Izmir  0.65  1.51  0.38    12.89        0.57
3  Finland          Helsinki  0.96  2.02  2.27    22.31           1
4  Moldova          Chisinau  0.68  1.11  0.33     8.58        0.18

  Electricity and Water
0                102.17
1                 59.33
2                 51.07
3                 82.66
4                113.46
```

[46]:
```
x.dtypes
```

[46]:
```
Country                  object
City                     object
Milk                     object
Eggs                     object
Bread                    object
Internet                 object
```

```
Taxi Travel              object
Electricity and Water    object
dtype: object
```

[47]:
```
x[['Milk','Bread','Eggs','Internet','Taxi Travel','Electricity and Water']].
 ↪astype('float').describe()
```

[47]:
```
             Milk       Bread        Eggs    Internet  Taxi Travel  \
count  160.000000  160.000000  160.000000  160.000000   160.000000
mean     0.998938    1.197875    1.902812   29.660875     0.922250
std      0.391720    0.760670    0.752520   18.908249     0.709011
min      0.390000    0.100000    0.750000    4.440000     0.140000
25%      0.710000    0.555000    1.377500   12.832500     0.405000
50%      0.895000    1.020000    1.850000   26.615000     0.630000
75%      1.170000    1.690000    2.352500   43.317500     1.350000
max      2.640000    3.330000    5.330000   93.290000     4.160000

       Electricity and Water
count             160.000000
mean              107.106125
std                51.553830
min                18.560000
25%                63.860000
50%               102.465000
75%               145.707500
max               265.520000
```

[48]:
```
plt.rcParams['figure.figsize'] = (10, 3)
# lets check those Countries where Milk is very Expensive
print(x[x['Milk'] > 1.17][['Country','City','Milk']].sort_values(by = 'Milk',
                                    ascending = False).head(5).
 ↪set_index('Country'))

print('\n')
# lets check those Countries where Bread is very Expensive
print(x[x['Bread'] > 1.69][['Country','City','Bread']].sort_values(by = 'Bread',
                                    ascending = False).head(5).
 ↪set_index('Country'))

print('\n')
# lets check those Countries where Bread is very Expensive
print(x[x['Eggs'] > 2.35][['Country','City','Eggs']].sort_values(by = 'Eggs',
                                    ascending = False).head(5).
 ↪set_index('Country'))

print('\n')
# lets check those Countries where Bread is very Expensive
```

```
print(x[x['Internet'] > 43.37][['Country','City','Internet']].sort_values(by =␣
 ↪'Internet',
                                    ascending = False).head(5).
 ↪set_index('Country'))

print('\n')
# lets check those Countries where Bread is very Expensive
print(x[x['Taxi Travel'] > 1.35][['Country','City','Taxi Travel']].
 ↪sort_values(by = 'Taxi Travel',
                                    ascending = False).head(5).
 ↪set_index('Country'))

print('\n')
# lets check those Countries where Bread is very Expensive
print(x[x['Electricity and Water'] > 145.7][['Country','City',
                        'Electricity and Water']].sort_values(by = 'Electricity␣
 ↪and Water',
                                    ascending = False).head(5).
 ↪set_index('Country'))
```

```
                  City  Milk
Country
Taiwan          Taipei  2.64
Hong Kong    Hong Kong  2.54
China         Shanghai  2.39
Singapore    Singapore  2.04
South Korea      Seoul  1.95


                  City Bread
Country
NY            New York  3.33
CA           San Diego  3.27
CA       San Francisco  3.12
CA         Los Angeles  2.99
Norway            Oslo  2.92


                  City  Eggs
Country
Switzerland     Zurich  5.33
Iceland      Reykjavik   4.8
Norway            Oslo  3.79
France           Paris   3.4
Israel       Jerusalem  3.32
```

```
                                  City  Internet
Country
United Arab Emirates  Abu Dhabi     93.29
United Arab Emirates      Dubai     90.42
Qatar                      Doha     78.31
AZ                      Phoenix     67.23
Costa Rica             San Jose      65.3
```

```
                                    City Taxi Travel
Country
Switzerland                       Zurich        4.16
Japan                              Tokyo        3.44
Netherlands                    Eindhoven           3
United Kingdom                    London        2.97
Dominican Republic         Santo Domingo        2.55
```

```
               City Electricity and Water
Country
Germany   Frankfurt                265.52
Germany      Munich                242.66
Germany     Hamburg                232.62
Germany      Berlin                 231.8
Slovenia  Ljubljana                199.61
```

0.2.2 Analyzing Quality of Life

[49]:
```python
life = pd.read_csv('movehubqualityoflife.csv')
life.head()
```

[49]:
```
           City  Movehub Rating  Purchase Power  Health Care  Pollution  \
0       Caracas           65.18           11.25        44.44      83.45
1  Johannesburg           84.08           53.99        59.98      47.39
2     Fortaleza           80.17           52.28        45.46      66.32
3   Saint Louis           85.25           80.40        77.29      31.33
4   Mexico City           75.07           24.28        61.76      18.95

   Quality of Life  Crime Rating
0             8.61         85.70
1            51.26         83.93
2            36.68         78.65
3            87.51         78.13
4            27.91         77.86
```

[50]:
```python
# analyzing the factors describing quality of life
life.describe()
```

[50]:

```
       Movehub Rating  Purchase Power  Health Care   Pollution  \
count      216.000000      216.000000   216.000000  216.000000
mean        79.676713       46.477176    66.442824   45.240370
std          6.501011       20.614519    14.416412   25.369741
min         59.880000        6.380000    20.830000    0.000000
25%         75.070000       28.815000    59.420000   24.410000
50%         81.060000       49.220000    67.685000   37.210000
75%         84.020000       61.607500    77.207500   67.675000
max        100.000000       91.850000    95.960000   92.420000

       Quality of Life  Crime Rating
count       216.000000    216.000000
mean         59.994537     41.338611
std          22.019376     16.416409
min           5.290000      9.110000
25%          42.752500     29.375000
50%          65.150000     41.140000
75%          78.617500     51.327500
max          97.910000     85.700000
```

[51]:

```
# lets analyze the Quality of Life

print('Cities having Best Quality of life')
display(life[['City','Quality of Life']].sort_values(by = 'Quality of Life',
                            ascending = False).head(10).set_index('City').style.
 ↪background_gradient(cmap = 'Reds'))

print('Cities having Worst Quality of life')
display(life[['City','Quality of Life']].sort_values(by = 'Quality of Life',
                            ascending = True).head(10).set_index('City').style.
 ↪background_gradient(cmap = 'Reds'))
```

```
Cities having Best Quality of life

<pandas.io.formats.style.Styler at 0xb5dbfd0>

Cities having Worst Quality of life

<pandas.io.formats.style.Styler at 0xb5dbc28>
```

[52]:

```
# lets analyze the heath care of cities

print('Cities having Best Health care Facility')
display(life[['City','Health Care']].sort_values(by = 'Health Care',
                            ascending = False).head(10).set_index('City').style.
 ↪background_gradient(cmap = 'Greens'))

print('Cities having Worst Health care Facility')
display(life[['City', 'Health Care']].sort_values(by = 'Health Care',
```

```
                            ascending = True).head(10).set_index('City').style.
↪background_gradient(cmap = 'Greens'))
```

```
Cities having Best Health care Facility
```

```
<pandas.io.formats.style.Styler at 0xb5addd8>
```

```
Cities having Worst Health care Facility
```

```
<pandas.io.formats.style.Styler at 0xb5ade68>
```

[53]:
```
# lets analyze the Crime Rate

print('Cities having Highest Crime Rate')
display(life[['City','Crime Rating']].sort_values(by = 'Crime Rating',
                            ascending = False).head(10).set_index('City').style.
↪background_gradient(cmap = 'bone'))

print('Cities having Worst Health care Facility')
display(life[['City', 'Crime Rating']].sort_values(by = 'Crime Rating',
                            ascending = True).head(10).set_index('City').style.
↪background_gradient(cmap = 'bone'))
```

```
Cities having Highest Crime Rate
```

```
<pandas.io.formats.style.Styler at 0xa48f2c8>
```

```
Cities having Worst Health care Facility
```

```
<pandas.io.formats.style.Styler at 0xb5b10a0>
```

0.3 Recommending Better Cities to live

[66]:
```
def recommend_better_cities(citi, factor = 'Lifestyle'):
    x = city[['City','Food','Education','Lifestyle','Travel', 'Income']]
    food = x[x['City'] == citi]['Food']
    edu = x[x['City'] == citi]['Education']
    life = x[x['City'] == citi]['Lifestyle']
    travel = x[x['City'] == citi]['Travel']
    income = x[x['City'] == citi]['Income']
    best_cities = x[(x['Food'] <= food.values[0]) & (x['Education'] <= edu.
↪values[0]) &
                  (x['Lifestyle'] <= life.values[0]) & (x['Travel'] <= travel.
↪values[0]) &
                  (x['Income'] > income.values[0])]
    best = best_cities.sort_values(by = factor, ascending = False).head(10)
    return best['City'].reset_index(drop = True)
```

[72]:
```
recommend_better_cities('Cape Town')
```

[72]:
```
0       Cape Town
1          Prague
2         Vilnius
3     Panama City
4          Durban
5          Kaunas
6          Warsaw
7          Gdansk
8            Brno
9          Poznan
Name: City, dtype: object
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: