In [1]:
```
import pandas as pd
data=pd.read_csv("C:/Users/TAPAN/Downloads/food_coded.csv")
```

In [2]:
```
data
```

Out[2]:

	GPA	Gender	breakfast	calories_chicken	calories_day	calories_scone	coffee	comfort_food	com
0	2.4	2	1	430	NaN	315.0	1	none	we
1	3.654	1	1	610	3.0	420.0	2	chocolate, chips, ice cream	
2	3.3	1	1	720	4.0	420.0	2	frozen yogurt, pizza, fast food	
3	3.2	1	1	430	3.0	420.0	2	Pizza, Mac and cheese, ice cream	
4	3.5	1	1	720	2.0	420.0	2	Ice cream, chocolate, chips	
...	...	...	...	...	...	...	...	...	
120	3.5	1	1	610	4.0	420.0	2	wine. mac and cheese, pizza, ice cream	bc
121	3	1	1	265	2.0	315.0	2	Pizza / Wings / Cheesecake	Lon
122	3.882	1	1	720	NaN	420.0	1	rice, potato, seaweed soup	
123	3	2	1	720	4.0	420.0	1	Mac n Cheese, Lasagna, Pizza	
124	3.9	1	1	430	NaN	315.0	2	Chocolates, pizza, and Ritz.	

125 rows × 61 columns

In [3]:
```
data.columns
```

Out[3]:
```
Index(['GPA', 'Gender', 'breakfast', 'calories_chicken', 'calories_day',
       'calories_scone', 'coffee', 'comfort_food', 'comfort_food_reasons',
       'comfort_food_reasons_coded', 'cook', 'comfort_food_reasons_coded.1',
       'cuisine', 'diet_current', 'diet_current_coded', 'drink',
       'eating_changes', 'eating_changes_coded', 'eating_changes_coded1',
       'eating_out', 'employment', 'ethnic_food', 'exercise',
       'father_education', 'father_profession', 'fav_cuisine',
```

```
       'fav_cuisine_coded', 'fav_food', 'food_childhood', 'fries', 'fruit_day',
       'grade_level', 'greek_food', 'healthy_feeling', 'healthy_meal',
       'ideal_diet', 'ideal_diet_coded', 'income', 'indian_food',
       'italian_food', 'life_rewarding', 'marital_status',
       'meals_dinner_friend', 'mother_education', 'mother_profession',
       'nutritional_check', 'on_off_campus', 'parents_cook', 'pay_meal_out',
       'persian_food', 'self_perception_weight', 'soup', 'sports', 'thai_food',
       'tortilla_calories', 'turkey_calories', 'type_sports', 'veggies_day',
       'vitamins', 'waffle_calories', 'weight'],
      dtype='object')
```

In [4]:
```
column=['cook','eating_out','employment','ethnic_food', 'exercise','fruit_day','income'
```

In [5]:
```
d=data[column]
```

In [6]:
```
d
```

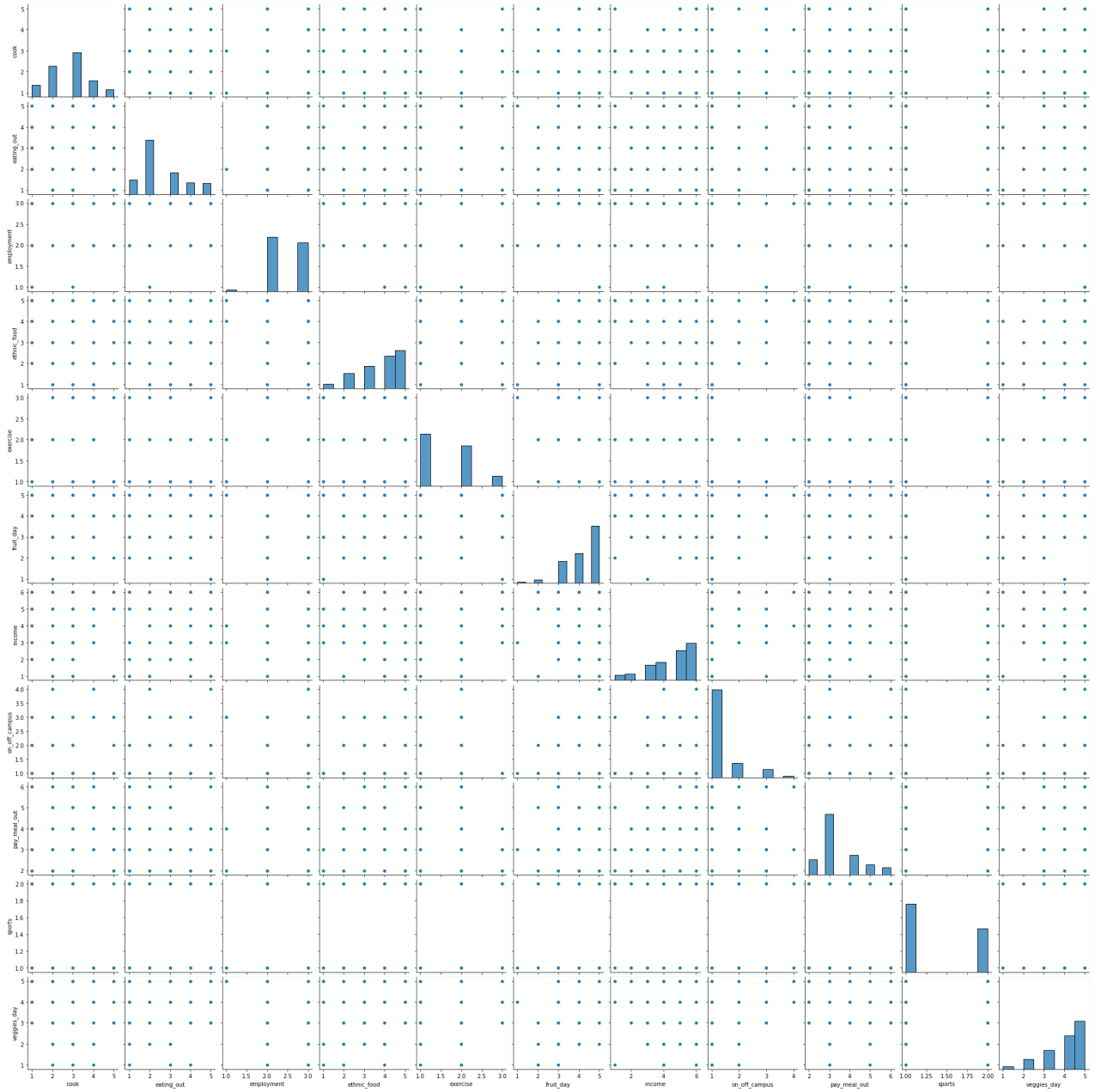
Out[6]:

	cook	eating_out	employment	ethnic_food	exercise	fruit_day	income	on_off_campus	pay_mea
0	2.0	3	3.0	1	1.0	5	5.0	1.0	
1	3.0	2	2.0	4	1.0	4	4.0	1.0	
2	1.0	2	3.0	5	2.0	5	6.0	2.0	
3	2.0	2	3.0	5	3.0	4	6.0	1.0	
4	1.0	2	2.0	4	1.0	4	6.0	1.0	
...	...	...	...	...	...	...	...	...	
120	3.0	2	1.0	4	2.0	5	4.0	3.0	
121	3.0	4	3.0	3	2.0	4	2.0	1.0	
122	3.0	3	3.0	5	2.0	4	2.0	1.0	
123	3.0	5	2.0	2	1.0	5	4.0	1.0	
124	NaN	1	2.0	3	2.0	3	5.0	1.0	

125 rows × 11 columns

In [7]:
```
import seaborn as sns
sns.pairplot(d)
```

Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x1848bf1ac10>
```

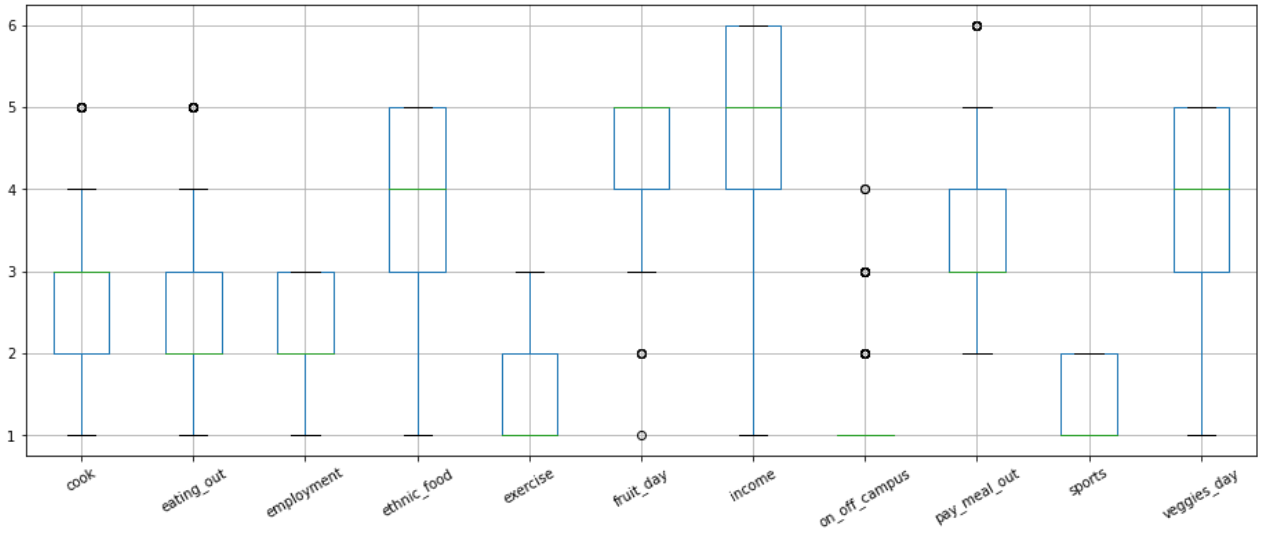


In [9]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
ax=d.boxplot(figsize=(16,6))
ax.set_xticklabels(ax.get_xticklabels(),rotation=30)
```

Out[9]:

```
[Text(1, 0, 'cook'),
 Text(2, 0, 'eating_out'),
 Text(3, 0, 'employment'),
 Text(4, 0, 'ethnic_food'),
 Text(5, 0, 'exercise'),
 Text(6, 0, 'fruit_day'),
 Text(7, 0, 'income'),
 Text(8, 0, 'on_off_campus'),
 Text(9, 0, 'pay_meal_out'),
 Text(10, 0, 'sports'),
 Text(11, 0, 'veggies_day')]
```



In [10]:
```
d.shape
```

Out[10]:
```
(125, 11)
```

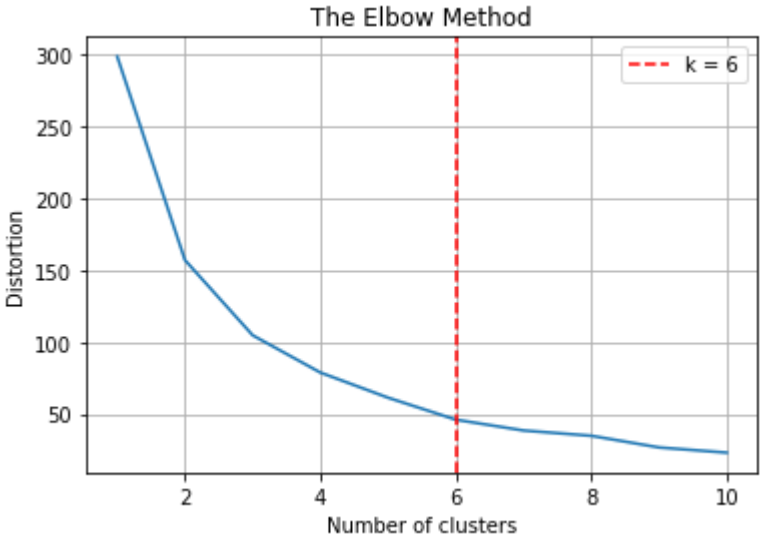
In [11]:
```
s=d.dropna()
```

In [12]:
```
import folium
import geopy
from sklearn import preprocessing, cluster
import scipy
import minisom
```

In [13]:
```
f=['cook','income']
X = s[f]
max_k = 10
## iterations
distortions = []
for i in range(1, max_k+1):
    if len(X) >= i:
       model = cluster.KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10,
       model.fit(X)
       distortions.append(model.inertia_)
## best k: the lowest derivative
k = [i*100 for i in np.diff(distortions,2)].index(min([i*100 for i
     in np.diff(distortions,2)]))
## plot
fig, ax = plt.subplots()
ax.plot(range(1, len(distortions)+1), distortions)
ax.axvline(k, ls='--', color="red", label="k = "+str(k))
ax.set(title='The Elbow Method', xlabel='Number of clusters',
       ylabel="Distortion")
ax.legend()
ax.grid(True)
plt.show()
```

```
C:\Users\TAPAN\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881: UserWarning:
KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
```

```
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=1.
  warnings.warn(
```



In [14]:
```
from pandas.io.json import json_normalize
import folium
from geopy.geocoders import Nominatim
import requests
CLIENT_ID = "KTCJJ2YZ2143QHEZ2JAQS4FJIO5DLSDO0YN4YBXPMI5NKTEF" # your Foursquare ID
CLIENT_SECRET = "KNG2LO22BPLHN1E3OAHWLYQ5PQBN14XYZMEMAS0CPJEJKOTR" # your Foursquare Se
VERSION = '20200316'
LIMIT = 10000
```

In [15]:
```
url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    17.448372, 78.526957,
    30000,
    LIMIT)
```

In [16]:
```
results = requests.get(url).json()
```

In [17]:
```
venues = results['response']['groups'][0]['items']
nearby_venues = json_normalize(venues)
```

```
C:\Users\TAPAN\AppData\Local\Temp/ipykernel_11440/2768318480.py:2: FutureWarning: panda
s.io.json.json_normalize is deprecated, use pandas.json_normalize instead
  nearby_venues = json_normalize(venues)
```

In [18]:
```
nearby_venues
```

Out[18]:

	referralId	reasons.count	reasons.items	venue.id	venue.name	v

	referralId	reasons.count	reasons.items	venue.id	venue.name
0	e-0-4c1f7229b306c928046b68b7-0	0	[{'summary': 'This spot is popular', 'type': '...	4c1f7229b306c928046b68b7	Fifth Avenue Bakers
1	e-0-5050c114e4b0694f643d178e-1	0	[{'summary': 'This spot is popular', 'type': '...	5050c114e4b0694f643d178e	Mekong
2	e-0-4ce690beb9975481a0faf044-2	0	[{'summary': 'This spot is popular', 'type': '...	4ce690beb9975481a0faf044	Chutneys
3	e-0-4df9c65c62e1e9a24367f9e5-3	0	[{'summary': 'This spot is popular', 'type': '...	4df9c65c62e1e9a24367f9e5	King & Cardinal
4	e-0-55e9d8dc498e8a5c51f30331-4	0	[{'summary': 'This spot is popular', 'type': '...	55e9d8dc498e8a5c51f30331	Cinepolis CCPL
...	...	...	...	...	...
95	e-0-4db439aa4df05e5aaadb21e6-95	0	[{'summary': 'This spot is popular', 'type': '...	4db439aa4df05e5aaadb21e6	Cafe Coffee Day
96	e-0-4ff37a1ee4b01d081ec95995-96	0	[{'summary': 'This spot is popular', 'type': '...	4ff37a1ee4b01d081ec95995	Karachi Bakery
97	e-0-4f8d4357e4b079c5bb18684a-97	0	[{'summary': 'This spot is popular', 'type': '...	4f8d4357e4b079c5bb18684a	Bawarchi
98	e-0-4cd80bd6da85224bf9764aca-98	0	[{'summary': 'This spot is popular', 'type': '...	4cd80bd6da85224bf9764aca	Pizza Hut
99	e-0-51682f3fe4b0c86be4c2d508-99	0	[{'summary': 'This spot is popular', 'type': '...	51682f3fe4b0c86be4c2d508	Matam Al-Arabi

100 rows × 22 columns

In [19]:

```
resta=[]
oth=[]
for lat,long in zip(nearby_venues['venue.location.lat'],nearby_venues['venue.location.l
    url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&
```

```
      CLIENT_ID,
      CLIENT_SECRET,
      VERSION,
      lat,long,
      1000,
      100)
    res = requests.get(url).json()
    venue = res['response']['groups'][0]['items']
    nearby_venue = json_normalize(venue)
    df=nearby_venue['venue.categories']

    g=[]
    for i in range(0,df.size):
      g.append(df[i][0]['icon']['prefix'].find('food'))
    co=0
    for i in g:
      if i>1:
        co+=1
    resta.append(co)
    oth.append(len(g)-co)

nearby_venues['restaurant']=resta
nearby_venues['others']=oth
nearby_venues
```

```
C:\Users\TAPAN\AppData\Local\Temp/ipykernel_11440/60030951.py:13: FutureWarning: pandas.
io.json.json_normalize is deprecated, use pandas.json_normalize instead
  nearby_venue = json_normalize(venue)
```

Out[19]:

	referralId	reasons.count	reasons.items	venue.id	venue.name	v
0	e-0-4c1f7229b306c928046b68b7-0	0	[{'summary': 'This spot is popular', 'type': '...	4c1f7229b306c928046b68b7	Fifth Avenue Bakers	
1	e-0-5050c114e4b0694f643d178e-1	0	[{'summary': 'This spot is popular', 'type': '...	5050c114e4b0694f643d178e	Mekong	
2	e-0-4ce690beb9975481a0faf044-2	0	[{'summary': 'This spot is popular', 'type': '...	4ce690beb9975481a0faf044	Chutneys	
3	e-0-4df9c65c62e1e9a24367f9e5-3	0	[{'summary': 'This spot is popular', 'type': '...	4df9c65c62e1e9a24367f9e5	King & Cardinal	
4	e-0-55e9d8dc498e8a5c51f30331-4	0	[{'summary': 'This spot is popular', 'type': '...	55e9d8dc498e8a5c51f30331	Cinepolis CCPL	
...	...	...	...	...	...	
95	e-0-4db439aa4df05e5aaadb21e6-95	0	[{'summary': 'This spot is popular', 'type': '...	4db439aa4df05e5aaadb21e6	Cafe Coffee Day	

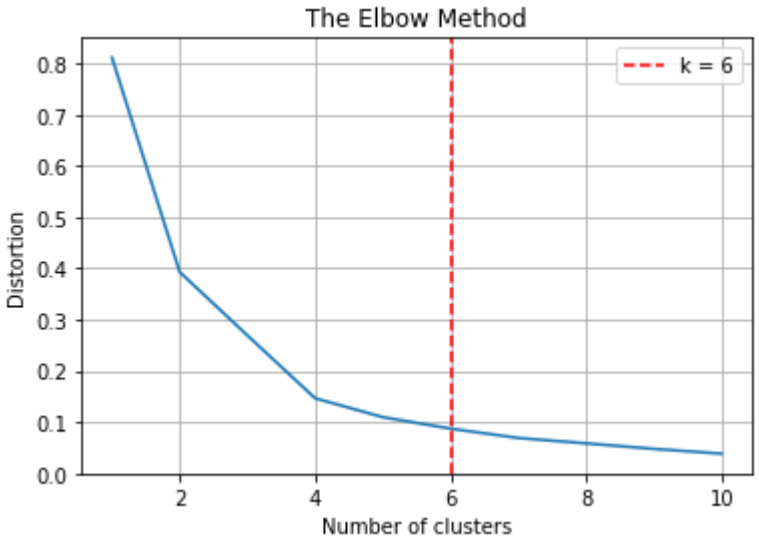
	referralId	reasons.count	reasons.items	venue.id	venue.name
96	e-0-4ff37a1ee4b01d081ec95995-96	0	[{'summary': 'This spot is popular', 'type': '...	4ff37a1ee4b01d081ec95995	Karachi Bakery
97	e-0-4f8d4357e4b079c5bb18684a-97	0	[{'summary': 'This spot is popular', 'type': '...	4f8d4357e4b079c5bb18684a	Bawarchi
98	e-0-4cd80bd6da85224bf9764aca-98	0	[{'summary': 'This spot is popular', 'type': '...	4cd80bd6da85224bf9764aca	Pizza Hut
99	e-0-51682f3fe4b0c86be4c2d508-99	0	[{'summary': 'This spot is popular', 'type': '...	51682f3fe4b0c86be4c2d508	Matam Al-Arabi

100 rows × 24 columns

In [20]:
```
lat=nearby_venues['venue.location.lat']
long=nearby_venues['venue.location.lng']
```

In [21]:
```
f=['venue.location.lat','venue.location.lng']
X = nearby_venues[f]
max_k = 10
## iterations
distortions = []
for i in range(1, max_k+1):
    if len(X) >= i:
       model = cluster.KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10,
       model.fit(X)
       distortions.append(model.inertia_)
## best k: the lowest derivative
k = [i*100 for i in np.diff(distortions,2)].index(min([i*100 for i
     in np.diff(distortions,2)]))
## plot
fig, ax = plt.subplots()
ax.plot(range(1, len(distortions)+1), distortions)
ax.axvline(k, ls='--', color="red", label="k = "+str(k))
ax.set(title='The Elbow Method', xlabel='Number of clusters',
       ylabel="Distortion")
ax.legend()
ax.grid(True)
plt.show()
```

```
C:\Users\TAPAN\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881: UserWarning:
KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=1.
  warnings.warn(
```



In [22]:
```
city = "Hyderabad"
## get location
locator = geopy.geocoders.Nominatim(user_agent="MyCoder")
location = locator.geocode(city)
print(location)
## keep latitude and longitude only
location = [location.latitude, location.longitude]
print("[lat, long]:", location)
```

```
Hyderabad, Bahadurpura mandal, Hyderabad, Telangana, India
[lat, long]: [17.360589, 78.4740613]
```

In [23]:
```
nearby_venues.head()
```

Out[23]:

	referralId	reasons.count	reasons.items	venue.id	venue.name	ve
0	e-0-4c1f7229b306c928046b68b7-0	0	[{'summary': 'This spot is popular', 'type': '...	4c1f7229b306c928046b68b7	Fifth Avenue Bakers	
1	e-0-5050c114e4b0694f643d178e-1	0	[{'summary': 'This spot is popular', 'type': '...	5050c114e4b0694f643d178e	Mekong	
2	e-0-4ce690beb9975481a0faf044-2	0	[{'summary': 'This spot is popular', 'type': '...	4ce690beb9975481a0faf044	Chutneys	
3	e-0-4df9c65c62e1e9a24367f9e5-3	0	[{'summary': 'This spot is popular', 'type': '...	4df9c65c62e1e9a24367f9e5	King & Cardinal	
4	e-0-55e9d8dc498e8a5c51f30331-4	0	[{'summary': 'This spot is popular', 'type': '...	55e9d8dc498e8a5c51f30331	Cinepolis CCPL	

5 rows × 24 columns

In [24]:
```
nearby_venues.columns
```

Out[24]:
```
Index(['referralId', 'reasons.count', 'reasons.items', 'venue.id',
       'venue.name', 'venue.location.address', 'venue.location.lat',
       'venue.location.lng', 'venue.location.labeledLatLngs',
       'venue.location.distance', 'venue.location.cc', 'venue.location.state',
       'venue.location.country', 'venue.location.formattedAddress',
       'venue.categories', 'venue.photos.count', 'venue.photos.groups',
       'venue.location.crossStreet', 'venue.location.postalCode',
       'venue.location.city', 'venue.venuePage.id',
       'venue.location.neighborhood', 'restaurant', 'others'],
      dtype='object')
```

In [25]:
```
n=nearby_venues.drop(['referralId', 'reasons.count', 'reasons.items', 'venue.id',
       'venue.name',
       'venue.location.labeledLatLngs', 'venue.location.distance',
       'venue.location.cc',
       'venue.categories', 'venue.photos.count', 'venue.photos.groups',
       'venue.location.crossStreet', 'venue.location.address','venue.location.city',
       'venue.location.state', 'venue.location.crossStreet',
       'venue.location.neighborhood',   'venue.venuePage.id',
       'venue.location.postalCode','venue.location.country'],axis=1)
```

In [26]:
```
n.columns
```

Out[26]:
```
Index(['venue.location.lat', 'venue.location.lng',
       'venue.location.formattedAddress', 'restaurant', 'others'],
      dtype='object')
```

In [27]:
```
n
```

Out[27]:

	venue.location.lat	venue.location.lng	venue.location.formattedAddress	restaurant	others
0	17.487673	78.542793	[Sainikpuri, Andhra Pradesh, India]	4	1
1	17.437151	78.454301	[Leelanagar (Begumpet), Hyderabad 500016, Tela...	37	23
2	17.443384	78.479939	[Sardar Patel Road (Adjacent to FedEx Centre),...	18	4
3	17.400678	78.488575	[Himayatnagar (Narayanguda-himayat Nagar X Roa...	21	8
4	17.457282	78.536823	[Malkajgiri, Hyderabad, Telangana, India]	6	6
...	...	...	...	...	...
95	17.330914	78.567641	[vanasthalipuram main Rd, Prasanthi Nagar, Van...	4	3
96	17.235138	78.430288	[Domestic Depatures (Shamshabad International ...	14	15

	venue.location.lat	venue.location.lng	venue.location.formattedAddress	restaurant	others
97	17.260920	78.386497	[India]	4	1
98	17.236604	78.429977	[Rajiv Gandhi International Airport, Opp KFC (...	14	15
99	17.301213	78.475706	[Barakas, India]	5	1

100 rows × 5 columns

In [28]:
```
n=n.dropna()
n = n.rename(columns={'venue.location.lat': 'lat', 'venue.location.lng': 'long'})
n
```

Out[28]:

	lat	long	venue.location.formattedAddress	restaurant	others
0	17.487673	78.542793	[Sainikpuri, Andhra Pradesh, India]	4	1
1	17.437151	78.454301	[Leelanagar (Begumpet), Hyderabad 500016, Tela...	37	23
2	17.443384	78.479939	[Sardar Patel Road (Adjacent to FedEx Centre),...	18	4
3	17.400678	78.488575	[Himayatnagar (Narayanguda-himayat Nagar X Roa...	21	8
4	17.457282	78.536823	[Malkajgiri, Hyderabad, Telangana, India]	6	6
...	...	...	...	...	...
95	17.330914	78.567641	[vanasthalipuram main Rd, Prasanthi Nagar, Van...	4	3
96	17.235138	78.430288	[Domestic Depatures (Shamshabad International ...	14	15
97	17.260920	78.386497	[India]	4	1
98	17.236604	78.429977	[Rajiv Gandhi International Airport, Opp KFC (...	14	15
99	17.301213	78.475706	[Barakas, India]	5	1

100 rows × 5 columns

In [29]:
```
n['venue.location.formattedAddress']
```

Out[29]:
```
0                  [Sainikpuri, Andhra Pradesh, India]
1     [Leelanagar (Begumpet), Hyderabad 500016, Tela...
2     [Sardar Patel Road (Adjacent to FedEx Centre),...
3     [Himayatnagar (Narayanguda-himayat Nagar X Roa...
4             [Malkajgiri, Hyderabad, Telangana, India]
                            ...
95    [vanasthalipuram main Rd, Prasanthi Nagar, Van...
96    [Domestic Depatures (Shamshabad International ...
97                                               [India]
98    [Rajiv Gandhi International Airport, Opp KFC (...
99                                    [Barakas, India]
Name: venue.location.formattedAddress, Length: 100, dtype: object
```

In [30]:
```
spec_chars = ["[","]"]
for char in spec_chars:
```

```
  n['venue.location.formattedAddress'] = n['venue.location.formattedAddress'].astype(st
```

```
C:\Users\TAPAN\AppData\Local\Temp/ipykernel_11440/3350752478.py:3: FutureWarning: The de
fault value of regex will change from True to False in a future version. In addition, si
ngle character regular expressions will *not* be treated as literal strings when regex=T
rue.
  n['venue.location.formattedAddress'] = n['venue.location.formattedAddress'].astype(st
r).str.replace(char, ' ')
```

In [31]:

```
n
```

Out[31]:

	lat	long	venue.location.formattedAddress	restaurant	others
0	17.487673	78.542793	'Sainikpuri', 'Andhra Pradesh', 'India'	4	1
1	17.437151	78.454301	'Leelanagar (Begumpet)', 'Hyderabad 500016', ...	37	23
2	17.443384	78.479939	'Sardar Patel Road (Adjacent to FedEx Centre)...	18	4
3	17.400678	78.488575	'Himayatnagar (Narayanguda-himayat Nagar X Ro...	21	8
4	17.457282	78.536823	'Malkajgiri', 'Hyderabad', 'Telangana', 'India'	6	6
...	...	...	...	...	...
95	17.330914	78.567641	'vanasthalipuram main Rd, Prasanthi Nagar, Va...	4	3
96	17.235138	78.430288	'Domestic Depatures (Shamshabad International...	14	15
97	17.260920	78.386497	'India'	4	1
98	17.236604	78.429977	'Rajiv Gandhi International Airport, Opp KFC ...	14	15
99	17.301213	78.475706	'Barakas', 'India'	5	1

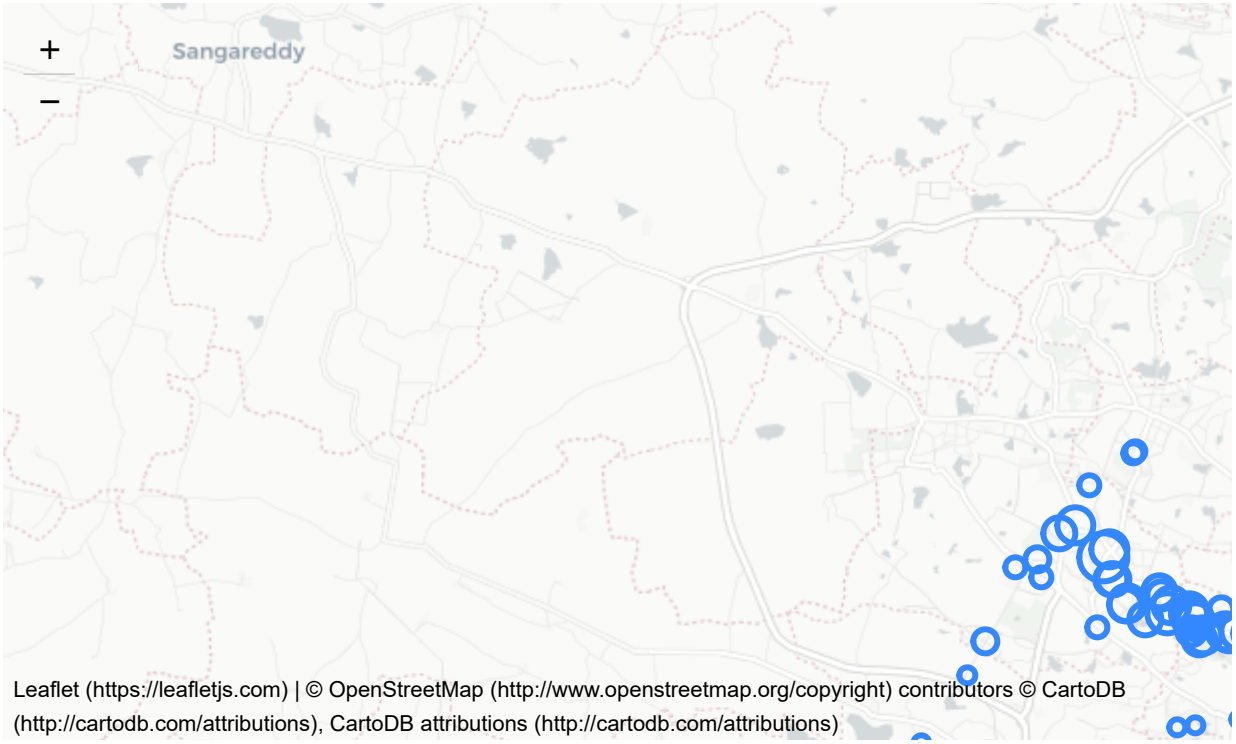
100 rows × 5 columns

In [32]:

```
x, y = "lat", "long"
color = "restaurant"
size = "others"
popup = "venue.location.formattedAddress"
data = n.copy()

## create color column
lst_colors=["red","green","orange"]
lst_elements = sorted(list(n[color].unique()))

## create size column (scaled)
scaler = preprocessing.MinMaxScaler(feature_range=(3,15))
data["size"] = scaler.fit_transform(
               data[size].values.reshape(-1,1)).reshape(-1)

## initialize the map with the starting location
map_ = folium.Map(location=location, tiles="cartodbpositron",
                  zoom_start=11)
## add points
data.apply(lambda row: folium.CircleMarker(
           location=[row[x],row[y]],popup=row[popup],
           radius=row["size"]).add_to(map_), axis=1)
```

```
## add html legend


## plot the map
map_
```

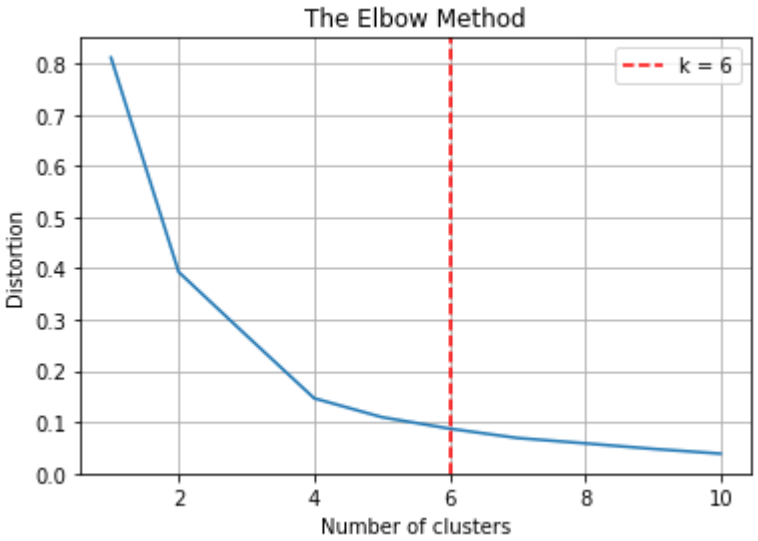
Out[32]:



In [33]:

```
X = n[["lat","long"]]
max_k = 10
## iterations
distortions = []
for i in range(1, max_k+1):
    if len(X) >= i:
       model = cluster.KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10,
       model.fit(X)
       distortions.append(model.inertia_)
## best k: the lowest derivative
k = [i*100 for i in np.diff(distortions,2)].index(min([i*100 for i in np.diff(distortio
## plot
fig, ax = plt.subplots()
ax.plot(range(1, len(distortions)+1), distortions)
ax.axvline(k, ls='--', color="red", label="k = "+str(k))
ax.set(title='The Elbow Method', xlabel='Number of clusters',
       ylabel="Distortion")
ax.legend()
ax.grid(True)
plt.show()
```

```
C:\Users\TAPAN\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881: UserWarning:
KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=1.
  warnings.warn(
```



In [34]:

```
k = 6
model = cluster.KMeans(n_clusters=k, init='k-means++')
X = n[["lat","long"]]
## clustering
dtf_X = X.copy()
dtf_X["cluster"] = model.fit_predict(X)
## find real centroids
closest, distances = scipy.cluster.vq.vq(model.cluster_centers_,
                     dtf_X.drop("cluster", axis=1).values)
dtf_X["centroids"] = 0
for i in closest:
    dtf_X["centroids"].iloc[i] = 1
## add clustering info to the original dataset
n[["cluster","centroids"]] = dtf_X[["cluster","centroids"]]
n
```

```
C:\Users\TAPAN\anaconda3\lib\site-packages\pandas\core\indexing.py:1732: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_block(indexer, value, name)
```

Out[34]:

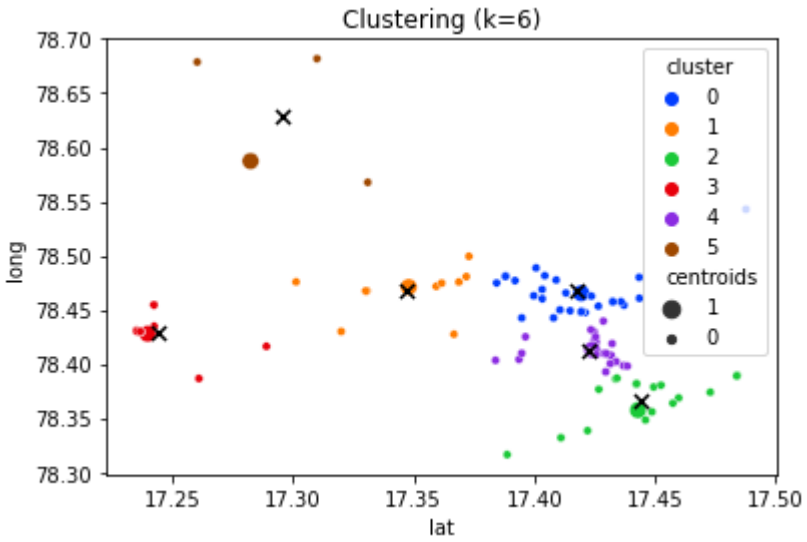
	lat	long	venue.location.formattedAddress	restaurant	others	cluster	centroids
0	17.487673	78.542793	'Sainikpuri', 'Andhra Pradesh', 'India'	4	1	0	0
1	17.437151	78.454301	'Leelanagar (Begumpet)', 'Hyderabad 500016', ...	37	23	0	0
2	17.443384	78.479939	'Sardar Patel Road (Adjacent to FedEx Centre)...	18	4	0	0
3	17.400678	78.488575	'Himayatnagar (Narayanguda-himayat Nagar X Ro...	21	8	0	0
4	17.457282	78.536823	'Malkajgiri', 'Hyderabad', 'Telangana', 'India'	6	6	0	0
...	...	...	...	...	...	...	...

	lat	long	venue.location.formattedAddress	restaurant	others	cluster	centroids
95	17.330914	78.567641	'vanasthalipuram main Rd, Prasanthi Nagar, Va...	4	3	5	0
96	17.235138	78.430288	'Domestic Depatures (Shamshabad International...	14	15	3	0
97	17.260920	78.386497	'India'	4	1	3	0
98	17.236604	78.429977	'Rajiv Gandhi International Airport, Opp KFC ...	14	15	3	0
99	17.301213	78.475706	'Barakas', 'India'	5	1	1	0

100 rows × 7 columns

In [35]:
```
fig, ax = plt.subplots()
sns.scatterplot(x="lat", y="long", data=n,
                palette=sns.color_palette("bright",k),
                hue='cluster', size="centroids", size_order=[1,0],
                legend="brief", ax=ax).set_title('Clustering (k='+str(k)+')')
th_centroids = model.cluster_centers_
ax.scatter(th_centroids[:,0], th_centroids[:,1], s=50, c='black',
           marker="x")
```

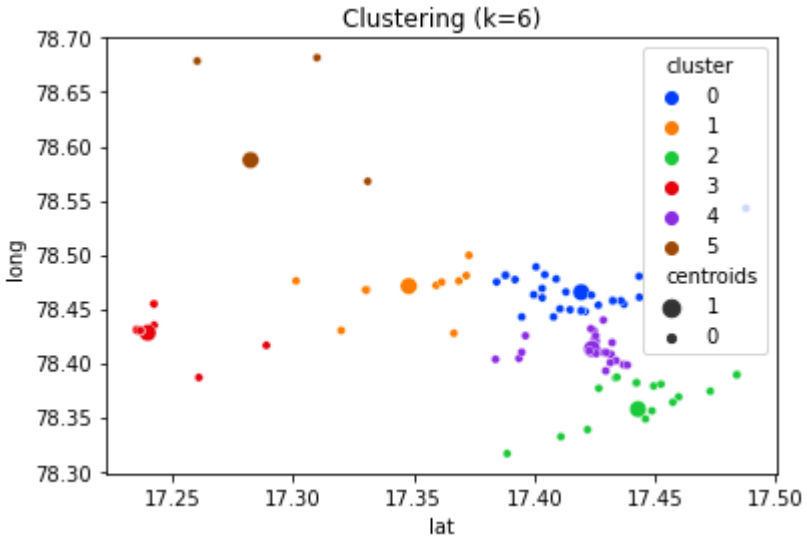
Out[35]:
```
<matplotlib.collections.PathCollection at 0x184988c5a30>
```



In [36]:
```
model = cluster.AffinityPropagation()
```

In [37]:
```
k = n["cluster"].nunique()
sns.scatterplot(x="lat", y="long", data=n,
                palette=sns.color_palette("bright",k),
                hue='cluster', size="centroids", size_order=[1,0],
                legend="brief").set_title('Clustering (k='+str(k)+')')
```

Out[37]:
```
Text(0.5, 1.0, 'Clustering (k=6)')
```



In [38]:

```
x, y = "lat", "long"
color = "cluster"
size = "restaurant"
popup = "venue.location.formattedAddress"
marker = "centroids"
data = n.copy()
## create color column
lst_elements = sorted(list(n[color].unique()))
lst_colors = ['#%06X' % np.random.randint(0, 0xFFFFFF) for i in
              range(len(lst_elements))]
data["color"] = data[color].apply(lambda x:
                lst_colors[lst_elements.index(x)])
## create size column (scaled)
scaler = preprocessing.MinMaxScaler(feature_range=(3,15))
data["size"] = scaler.fit_transform(
               data[size].values.reshape(-1,1)).reshape(-1)
## initialize the map with the starting location
map_ = folium.Map(location=location, tiles="cartodbpositron",
                  zoom_start=11)
## add points
data.apply(lambda row: folium.CircleMarker(
           location=[row[x],row[y]],
           color=row["color"], fill=True,popup=row[popup],
           radius=row["size"]).add_to(map_), axis=1)
## add html legend
legend_html = """ """+color+""":
"""
for i in lst_elements:
     legend_html = legend_html+"""      fa-1x" style="color:"""+lst_colors[lst_elements
      """+str(i)+"""
"""
legend_html = legend_html+"""""""
map_.get_root().html.add_child(folium.Element(legend_html))
## add centroids marker
lst_elements = sorted(list(n[marker].unique()))
data[data[marker]==1].apply(lambda row:
           folium.Marker(location=[row[x],row[y]],
           draggable=False,  popup=row[popup] ,
           icon=folium.Icon(color="black")).add_to(map_), axis=1)
## plot the map
map_
```

Out[38]:

cluster fa-1x" style="color:#CE74C3"> 0 fa-1x" style="color:#A08CFE"> 1 fa-1x" style="color:#BD0996"> 2 fa-1x" style="color:#0EE6B2"> 3 fa-1x" style="color:#76EB8B"> 4 fa-1x" style="color:#C1CD10"> 5

In [ ]: