hack

April 19, 2025

[5]:
```
!pip install --quiet geopy folium
```

[7]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


from geopy.geocoders import Nominatim
from geopy.exc import GeocoderTimedOut


from IPython.display import display, HTML


import time


import folium
```

[9]:
```
df = pd.read_csv('zomato_data.csv')


geo_df = pd.read_csv('Geographical Coordinates.csv')
```

[21]:
```
df['rate'] = df['rate'].replace(['-', 'NEW'], np.nan)


df['rate'] = df['rate'].astype(str)


df['rate'] = df['rate'].str.replace('/5', '', regex=False)


df['rate'] = pd.to_numeric(df['rate'], errors='coerce')
```

```
df['rate'] = df['rate'].fillna(df['rate'].median())
```

[25]:
```
df['approx_costfor_two_people'] = df['approx_costfor_two_people'].astype(str).
 ↪str.replace(',', '', regex=False)


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


df['approx_costfor_two_people'] = pd.
 ↪to_numeric(df['approx_costfor_two_people'], errors='coerce')


df['approx_costfor_two_people'] = df['approx_costfor_two_people'].
 ↪fillna(df['approx_costfor_two_people'].median())
```

[29]:
```
# Fill missing values in 'votes' with the median of the column
df['votes'] = df['votes'].fillna(df['votes'].median())
```

[31]:
```
df['online_order'] = df['online_order'].map({'Yes': 1, 'No': 0})
df['book_table'] = df['book_table'].map({'Yes': 1, 'No': 0})
```

[33]:
```
df['rate'] = df['rate'].astype(float)
df['votes'] = df['votes'].astype(int)
df['approx_costfor_two_people'] = df['approx_costfor_two_people'].astype(int)
```

[35]:
```
df.info()
df.isnull().sum()
df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51717 entries, 0 to 51716
Data columns (total 10 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   online_order                 51717 non-null  int64
 1   book_table                   51717 non-null  int64
 2   rate                         51717 non-null  float64
 3   votes                        51717 non-null  int32
 4   rest_type                    51490 non-null  object
 5   dish_liked                   23639 non-null  object
 6   cuisines                     51672 non-null  object
 7   approx_costfor_two_people    51717 non-null  int32
 8   listed_intype                51717 non-null  object
 9   listed_incity                51717 non-null  object
dtypes: float64(1), int32(2), int64(2), object(5)
```

```
memory usage: 3.6+ MB
```

[35]:
```
       online_order    book_table          rate         votes  \
count  51717.000000  51717.000000  51717.000000  51717.000000
mean       0.588665      0.124698      3.700362    283.697527
std        0.492080      0.330379      0.395391    803.838853
min        0.000000      0.000000      1.800000      0.000000
25%        0.000000      0.000000      3.500000      7.000000
50%        1.000000      0.000000      3.700000     41.000000
75%        1.000000      0.000000      3.900000    198.000000
max        1.000000      1.000000      4.900000  16832.000000

       approx_costfor_two_people
count                51717.000000
mean                   554.391689
std                    437.563723
min                     40.000000
25%                    300.000000
50%                    400.000000
75%                    650.000000
max                   6000.000000
```

[37]:
```
merged_df = pd.merge(df, geo_df, on='listed_incity', how='left')
merged_df.head()
```

[37]:
```
   online_order  book_table  rate  votes           rest_type  \
0             1           1   4.1    775      Casual Dining
1             1           0   4.1    787      Casual Dining
2             1           0   3.8    918  Cafe, Casual Dining
3             0           0   3.7     88        Quick Bites
4             0           0   3.8    166      Casual Dining

                                        dish_liked  \
0  Pasta, Lunch Buffet, Masala Papad, Paneer Laja…
1  Momos, Lunch Buffet, Chocolate Nirvana, Thai G…
2  Churros, Cannelloni, Minestrone Soup, Hot Choc…
3                                     Masala Dosa
4                             Panipuri, Gol Gappe

                       cuisines  approx_costfor_two_people listed_intype  \
0  North Indian, Mughlai, Chinese                        800        Buffet
1     Chinese, North Indian, Thai                        800        Buffet
2          Cafe, Mexican, Italian                        800        Buffet
3      South Indian, North Indian                        300        Buffet
4        North Indian, Rajasthani                        600        Buffet

  listed_incity   Latitude  Longitude
```

```
0  Banashankari  12.939333  77.553982
1  Banashankari  12.939333  77.553982
2  Banashankari  12.939333  77.553982
3  Banashankari  12.939333  77.553982
4  Banashankari  12.939333  77.553982
```

[39]:
```
bangalore_map = folium.Map(location=[12.9716, 77.5946], zoom_start=11)


for idx, row in merged_df.iterrows():
    if not pd.isnull(row['Latitude']) and not pd.isnull(row['Longitude']):
        folium.CircleMarker(
            location=[row['Latitude'], row['Longitude']],
            radius=1,
            color='blue',
            fill=True,
            fill_color='blue',
            fill_opacity=0.5
        ).add_to(bangalore_map)

bangalore_map
```

[39]:
```
<folium.folium.Map at 0x20079e0be00>
```

[41]:
```
italian_df = merged_df[merged_df['cuisines'].str.contains('Italian',␣
 ↪case=False, na=False)]

italian_map = folium.Map(location=[12.9716, 77.5946], zoom_start=11)

for idx, row in italian_df.iterrows():
    if not pd.isnull(row['Latitude']) and not pd.isnull(row['Longitude']):
        folium.Marker(
            location=[row['Latitude'], row['Longitude']],
            popup=row['name'],
            icon=folium.Icon(color='red', icon='cutlery', prefix='fa')
        ).add_to(italian_map)

italian_map
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3805, in Index.
 ↪get_loc(self, key)
   3804 try:
-> 3805     return self._engine.get_loc(casted_key)
   3806 except KeyError as err:
```

```
File index.pyx:167, in pandas._libs.index.IndexEngine.get_loc()

File index.pyx:196, in pandas._libs.index.IndexEngine.get_loc()

File pandas\\_libs\\hashtable_class_helper.pxi:7081, in pandas._libs.hashtable.
 ↪PyObjectHashTable.get_item()

File pandas\\_libs\\hashtable_class_helper.pxi:7089, in pandas._libs.hashtable.
 ↪PyObjectHashTable.get_item()

KeyError: 'name'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[41], line 9
      5 for idx, row in italian_df.iterrows():
      6     if not pd.isnull(row['Latitude']) and not pd.
 ↪isnull(row['Longitude']):
      7         folium.Marker(
      8             location=[row['Latitude'], row['Longitude']],
----> 9             popup=row['name'],
     10             icon=folium.Icon(color='red', icon='cutlery', prefix='fa')
     11         ).add_to(italian_map)
     13 italian_map

File ~\anaconda3\Lib\site-packages\pandas\core\series.py:1121, in Series.
 ↪__getitem__(self, key)
   1118     return self._values[key]
   1120 elif key_is_scalar:
-> 1121     return self._get_value(key)
   1123 # Convert generator to list before going through hashable part
   1124 # (We will iterate through the generator there to check for slices)
   1125 if is_iterator(key):

File ~\anaconda3\Lib\site-packages\pandas\core\series.py:1237, in Series.
 ↪_get_value(self, label, takeable)
   1234     return self._values[label]
   1236 # Similar to Index.get_value, but we do not fall back to positional
-> 1237 loc = self.index.get_loc(label)
   1239 if is_integer(loc):
   1240     return self._values[loc]

File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3812, in Index.
 ↪get_loc(self, key)
   3807     if isinstance(casted_key, slice) or (
   3808         isinstance(casted_key, abc.Iterable)
   3809         and any(isinstance(x, slice) for x in casted_key)
```

```
   3810     ):
   3811         raise InvalidIndexError(key)
-> 3812     raise KeyError(key) from err
   3813 except TypeError:
   3814     # If we have a listlike key, _check_indexing_error will raise
   3815     #  InvalidIndexError. Otherwise we fall through and re-raise
   3816     #  the TypeError.
   3817     self._check_indexing_error(key)

KeyError: 'name'
```

[43]:
```
print(italian_df.columns)
```

```
Index(['online_order', 'book_table', 'rate', 'votes', 'rest_type',
       'dish_liked', 'cuisines', 'approx_costfor_two_people', 'listed_intype',
       'listed_incity', 'Latitude', 'Longitude'],
      dtype='object')
```

[47]:
```
import folium
import pandas as pd


print("Italian DF columns:", italian_df.columns)


name_col = None
possible_name_cols = ['name', 'restaurant_name', 'Restaurant Name', 'res_name']
for col in possible_name_cols:
    if col in italian_df.columns:
        name_col = col
        break


if name_col is None:
    name_col = italian_df.columns[0]
    print("  Name column not found, using first column:", name_col)


bangalore_coords = [12.9716, 77.5946]
italian_map = folium.Map(location=bangalore_coords, zoom_start=12)


for idx, row in italian_df.iterrows():
    if pd.notnull(row['Latitude']) and pd.notnull(row['Longitude']):
        popup_text = str(row[name_col])
        folium.Marker(
            location=[row['Latitude'], row['Longitude']],
```

```
            popup=popup_text,
            icon=folium.Icon(color='red', icon='cutlery', prefix='fa')
        ).add_to(italian_map)


italian_map
```

```
Italian DF columns: Index(['online_order', 'book_table', 'rate', 'votes',
'rest_type',
       'dish_liked', 'cuisines', 'approx_costfor_two_people', 'listed_intype',
       'listed_incity', 'Latitude', 'Longitude'],
      dtype='object')
 Name column not found, using first column: online_order
```

[47]:
```
<folium.folium.Map at 0x2000c188f20>
```

[ ]:
```

```