ARATRIK ROY CHOUDHURY

aratrikrc10@gmail.com

+91 7001542366

Coreshield Technologies SDE-1 Shortlisting Assignment

Objective:

You have been provided with two JSON files:

● The first JSON contains location identifiers (id), latitude, and longitude.

● The second JSON contains additional metadata related to each location identifier.

Process and Analyze Data:

● Load parse both JSON files and merge the data based on the matching id.

● Count how many valid points exist per type (e.g., restaurants, hotels, cafes, museums, parks).

● Calculate the average rating per type, considering only valid entries.

● Identify the location with the highest number of reviews.

● Identify locations with incomplete data if any.(*Bonus point)

● Technical Stack : As per your preference.

SOLUTION:-

INTRO:

This document outlines the step-by-step approach to solve the assignment involving the processing and analysis of two JSON files. The goal is to merge the data, perform specific calculations, and extract insights. The technical stack used is Python, leveraging libraries such as **JSON, PANDAS, and NUMPY.**

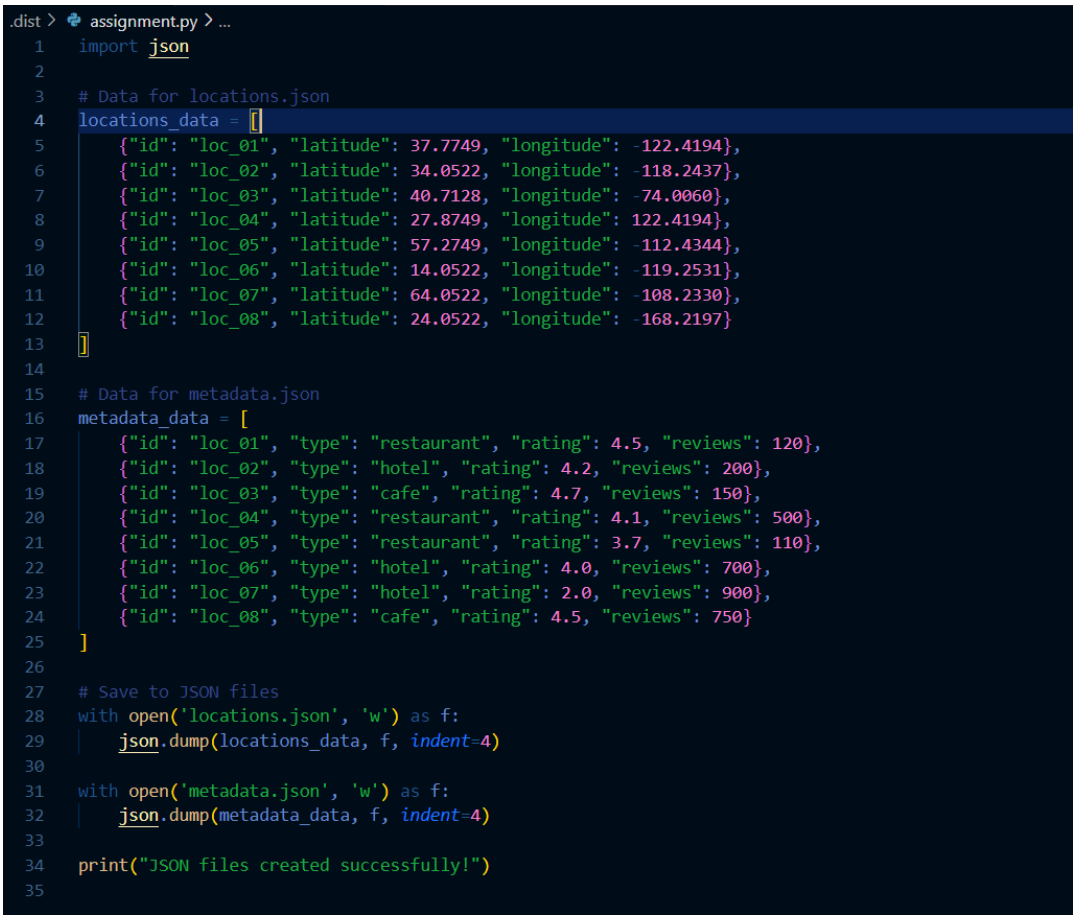
Approach:-

This assignment typically involves

- Merging 2 JSON files(Locations.json & Metadata.json) based on a common **id** field.
- Counting all the valid points per type (for example: cafes, restaurants & hotels).
- Next we move on to calculating the average rating per type.
- Once we are done with the Calculations we move on to identifying location with the highest number of reviews.
- As a part of our bonus objective we also filter out the locations with the incomplete data.

Step By Step Approach for better understanding:-

1. Create the Json files and then Load and Parse them Either in notepad or VScode
   We can do this either manually by entering json files in correct format with the help of notepad or vscode
   OR
   We can directly create both the JSON files with the hel of python libraries which I have used to make the process more smooth and hassle free directly in VS code



```
.dist > assignment.py > ...
1   import json
2
3   # Data for locations.json
4   locations_data = [
5       {"id": "loc_01", "latitude": 37.7749, "longitude": -122.4194},
6       {"id": "loc_02", "latitude": 34.0522, "longitude": -118.2437},
7       {"id": "loc_03", "latitude": 40.7128, "longitude": -74.0060},
8       {"id": "loc_04", "latitude": 27.8749, "longitude": 122.4194},
9       {"id": "loc_05", "latitude": 57.2749, "longitude": -112.4344},
10      {"id": "loc_06", "latitude": 14.0522, "longitude": -119.2531},
11      {"id": "loc_07", "latitude": 64.0522, "longitude": -108.2330},
12      {"id": "loc_08", "latitude": 24.0522, "longitude": -168.2197}
13  ]
14
15  # Data for metadata.json
16  metadata_data = [
17      {"id": "loc_01", "type": "restaurant", "rating": 4.5, "reviews": 120},
18      {"id": "loc_02", "type": "hotel", "rating": 4.2, "reviews": 200},
19      {"id": "loc_03", "type": "cafe", "rating": 4.7, "reviews": 150},
20      {"id": "loc_04", "type": "restaurant", "rating": 4.1, "reviews": 500},
21      {"id": "loc_05", "type": "restaurant", "rating": 3.7, "reviews": 110},
22      {"id": "loc_06", "type": "hotel", "rating": 4.0, "reviews": 700},
23      {"id": "loc_07", "type": "hotel", "rating": 2.0, "reviews": 900},
24      {"id": "loc_08", "type": "cafe", "rating": 4.5, "reviews": 750}
25  ]
26
27  # Save to JSON files
28  with open('locations.json', 'w') as f:
29      json.dump(locations_data, f, indent=4)
30
31  with open('metadata.json', 'w') as f:
32      json.dump(metadata_data, f, indent=4)
33
34  print("JSON files created successfully!")
35
```

Alternatively once created we can Load the JSON files into python and convert them into pandas DataFrames for easier manipulation .

Code:-

```
import json
import pandas as pd

# Load JSON files
with open('locations.json') as f:
    locations_data = json.load(f)

with open('metadata.json') as f:
    metadata_data = json.load(f)

# Convert to DataFrames
locations_df = pd.DataFrame(locations_data)
metadata_df = pd.DataFrame(metadata_data)
```

2. Once done with the validation and Loading of the Json files we move on to the merging stage based on ID:

   Here we combine the two Dataframes Using id column as the key .

   Code:-

```
valid_points_per_type = merged_df['type'].value_counts()
print("Valid Points per Type:\n", valid_points_per_type)
```

3. After merging both the JSON files we have our entire workbase ready with the data to work on based on the objectives thus we move on to the next phrase which is **Counting Valid Points per Type**

   Here we Count the number of valid entries for each type (example: cafes,hotels and restaurants).

   CODE:-

```
valid_points_per_type = merged_df['type'].value_counts()

print("Valid Points per Type:\n", valid_points_per_type)
```

```
=== Valid Points per Type ===
      Type  Count
restaurant      3
     hotel      3
      cafe      2
```

4. Next we need to maintain a record for Average Rating per Type
   Here we group the data by TYPE and calculate the average rating for each group

   CODE:-
   average_rating_per_type = merged_df.groupby('type')['rating'].mean()
   print("Average Rating per Type:\n", average_rating_per_type)

```
=== Average Rating per Type ===
       Type  Average Rating
       cafe             4.6
      hotel             3.4
 restaurant             4.1
```

5. Going onto the next step we need to identify the **Location with the Highest Number of Reviews**

   Here we find the location with the maximum value in the reviews column

   CODE:-
   incomplete_data = merged_df[merged_df.isnull().any(axis=1)]
   print("Locations with Incomplete Data:\n", incomplete_data)

```
=== Location with Highest Reviews ===
     id latitude longitude  type rating reviews
loc_07  64.0522  -108.233 hotel    2.0     900
```

6. Finally as a part of the Bonus we move on the last step which is **Identifying Locations with incomplete Data**

   Check for the Missing Values in the merged DataFrame.

   CODE:

```
incomplete_data = merged_df[merged_df.isnull().any(axis=1)]
print("Locations with Incomplete Data:\n", incomplete_data)
```

```
Locations with Incomplete Data:
 Empty DataFrame
Columns: [id, latitude, longitude, type, rating, reviews]
Index: []
```

```
=== Locations with Incomplete Data ===
No incomplete data found.
```

   COMPLETE CODE:-

```
import json
import pandas as pd

with open('locations.json') as f:
    locations_data = json.load(f)

with open('metadata.json') as f:
    metadata_data = json.load(f)

locations_df = pd.DataFrame(locations_data)
metadata_df = pd.DataFrame(metadata_data)

merged_df = pd.merge(locations_df, metadata_df, on='id', how='inner')

valid_points_per_type = merged_df['type'].value_counts().reset_index()
valid_points_per_type.columns = ['Type', 'Count']

average_rating_per_type = merged_df.groupby('type')['rating'].mean().reset_index()
average_rating_per_type.columns = ['Type', 'Average Rating']

location_with_max_reviews = merged_df.loc[merged_df['reviews'].idxmax()]
```

incomplete_data = merged_df[merged_df.isnull().any(*axis*=1)]

print("\n=== Valid Points per Type ===")
print(valid_points_per_type.to_string(*index*=False))

print("\n=== Average Rating per Type ===")
print(average_rating_per_type.to_string(*index*=False))

print("\n=== Location with Highest Reviews ===")
print(pd.DataFrame(location_with_max_reviews).T.to_string(*index*=False))

print("\n=== Locations with Incomplete Data ===")
if incomplete_data.empty:
    print("No incomplete data found.")
else:
    print(incomplete_data.to_string(*index*=False))

```
import json
import pandas as pd
with open('locations.json') as f:
    locations_data = json.load(f)
with open('metadata.json') as f:
    metadata_data = json.load(f)
locations_df = pd.DataFrame(locations_data)
metadata_df = pd.DataFrame(metadata_data)
merged_df = pd.merge(locations_df, metadata_df, on='id', how='inner')

valid_points_per_type = merged_df['type'].value_counts().reset_index()
valid_points_per_type.columns = ['Type', 'Count']

average_rating_per_type = merged_df.groupby('type')['rating'].mean().reset_index()
average_rating_per_type.columns = ['Type', 'Average Rating']

location_with_max_reviews = merged_df.loc[merged_df['reviews'].idxmax()]
incomplete_data = merged_df[merged_df.isnull().any(axis=1)]


print("\n=== Valid Points per Type ===")
print(valid_points_per_type.to_string(index=False))

print("\n=== Average Rating per Type ===")
print(average_rating_per_type.to_string(index=False))

print("\n=== Location with Highest Reviews ===")
print(pd.DataFrame(location_with_max_reviews).T.to_string(index=False))

print("\n=== Locations with Incomplete Data ===")
if incomplete_data.empty:
    print("No incomplete data found.")
else:
    print(incomplete_data.to_string(index=False))

```

OUTPUT:-

```
PS C:\Users\aratr\OneDrive\Desktop\coreshield assignment> & c:/Users/aratr/One
sers/aratr/OneDrive/Desktop/coreshield assignment/.dist/FinalOperation.py"

=== Valid Points per Type ===
       Type  Count
restaurant      3
     hotel      3
      cafe      2

=== Average Rating per Type ===
       Type  Average Rating
       cafe             4.6
     hotel              3.4
restaurant              4.1

=== Location with Highest Reviews ===
    id latitude longitude  type rating reviews
loc_07  64.0522  -108.233 hotel    2.0     900

=== Locations with Incomplete Data ===
No incomplete data found.
PS C:\Users\aratr\OneDrive\Desktop\coreshield assignment>
```

CONCLUSION:-

Thus this assignment was completed successfully by merging the two JSON files and performing the required calculations .The use of Python and pandas made the process efficient and scalable .The results provide valuable insights into the data ,such as the most reviewed location and the average ratings per type

FUTURE ENHANCEMENTS:-

- Handle missing data more robustly (for example fill or drop missing values).
- Visualize the results using libraries like matplotlib or seaborn.
- Extend the analysis to include additional metrics (for example-distance calculations).

GITHUB LINK : https://github.com/AratrikRC07/CoreShield-JSON-Assignment