* Read data from JSON files using Pandas
* Create Dataframe using JSON Array
* Process REST Payloads using Pandas
* Dealing with Complex JSON Data
* Overview of json\_normalize
* Write data in Dataframe to JSON files
* Convert CSV Files to JSON using Compression
* Exercise and Solution

1. How can you read data from a JSON file using Pandas?

A. pd.read\_json("filename.json")

B. pd.read\_csv("filename.json")

C. pd.read\_excel("filename.json")

D. pd.read\_txt("filename.json")

Answer: A. pd.read\_json("filename.json")

1. How can you create a DataFrame using a JSON array?

A. pd.read\_json(json\_array)

B. pd.DataFrame(json\_array)

C. pd.DataFrame.from\_json(json\_array)

D. pd.json\_to\_dataframe(json\_array)

Answer: B. pd.DataFrame(json\_array)

1. How can you process REST payloads using Pandas?

A. Use the requests library to fetch the data and then use Pandas to create a DataFrame.

B. Use Pandas to directly fetch the data from the REST API.

C. Use the Flask library to create a REST API and then use Pandas to process the data.

D. Use the Django library to create a REST API and then use Pandas to process the data.

Answer: A. Use the requests library to fetch the data and then use Pandas to create a DataFrame.

1. How can you deal with complex JSON data using Pandas?

A. Use the json\_normalize function to flatten the nested JSON data.

B. Use the pd.read\_json function with the "orient" parameter set to "records".

C. Use the pd.read\_json function with the "orient" parameter set to "columns".

D. Use the pd.DataFrame function with the "data" parameter set to the JSON data.

Answer: A. Use the json\_normalize function to flatten the nested JSON data.

1. What is the purpose of json\_normalize in Pandas?

A. To convert JSON data to a Pandas DataFrame.

B. To flatten nested JSON data into a DataFrame.

C. To transform a DataFrame into JSON data.

D. To merge two JSON files into a single DataFrame.

Answer: B. To flatten nested JSON data into a DataFrame.

1. How can you write data in a DataFrame to a JSON file using Pandas?

A. df.to\_json("filename.json")

B. df.write\_json("filename.json")

C. pd.write\_json(df, "filename.json")

D. pd.to\_json(df, "filename.json")

Answer: A. df.to\_json("filename.json")

1. How can you convert a CSV file to JSON using compression?

A. Use the pd.read\_csv function and then use the gzip library to compress the resulting DataFrame to a JSON file.

B. Use the pd.read\_csv function to read the CSV file into a DataFrame, use the df.to\_json function to convert the DataFrame to a JSON file, and then use the gzip library to compress the JSON file.

C. Use the pd.read\_csv function to read the CSV file into a DataFrame, use the df.to\_json function to convert the DataFrame to a JSON file, and then use the bz2 library to compress the JSON file.

D. Use the pd.read\_csv function to read the CSV file into a DataFrame, use the df.to\_json function to convert the DataFrame to a JSON file, and then use the zlib library to compress the JSON file.

Answer: D. Use the pd.read\_csv function to read the CSV file into a DataFrame, use the df.to\_json function to convert the DataFrame to a JSON file, and then use the zlib library to compress the JSON file.

1. Which of the following is an example of JSON data?

A. {"name": "John", "age": 30, "city": "New York"}

B. name: John, age: 30, city: New York

C. <person><name>John</name><age>30</age><city>New York</city></person>

D. ["John", "30", "New York"]

Answer: A. {"name": "John", "age": 30, "city": "New York"}

1. How can you create a DataFrame from a dictionary containing JSON data?

A. pd.DataFrame(json\_data)

B. pd.DataFrame.from\_json(json\_data)

C. pd.json\_to\_dataframe(json\_data)

D. pd.DataFrame.from\_dict(json\_data)

Answer: D. pd.DataFrame.from\_dict(json\_data)

1. Which of the following is an example of using json\_normalize to flatten nested JSON data?

A. json\_normalize(nested\_data)

B. pd.json\_normalize(nested\_data)

C. json\_normalize(nested\_data, "keys")

D. pd.json\_normalize(nested\_data, "values")

Answer: B. pd.json\_normalize(nested\_data)