Solutions

1. Excel spreadsheets have several advantages over CSV (Comma-Separated Values) spreadsheets:

   Formatting: Excel allows you to format your data in many ways, such as applying font styles and colors, changing cell backgrounds, and adding borders. This can make your data more visually appealing and easier to read and understand.

   Formulas and Functions: Excel has a vast library of built-in formulas and functions that can be used to perform calculations on your data, such as SUM, AVERAGE, and COUNT. This can save you a lot of time and effort compared to manually performing calculations in a CSV spreadsheet.

   Data Validation: Excel allows you to set rules for data entry in your spreadsheet, such as requiring certain fields to be filled in, or only allowing certain types of data to be entered. This can help ensure the accuracy and consistency of your data.

   Charts and Graphs: Excel has powerful charting and graphing capabilities that allow you to visualize your data in many ways, such as bar charts, line graphs, and pie charts. This can help you identify patterns and trends in your data more easily.

   Collaboration: Excel allows multiple users to work on the same spreadsheet at the same time, with changes being synced in real-time. This can make it easier for teams to collaborate on data analysis and reporting.

2.

```
import csv

# Open a CSV file for reading
with open('data.csv', 'r') as csv_file:
    # Create a reader object
    csv_reader = csv.reader(csv_file, delimiter=',',
quotechar='"')

    # Loop through each row in the CSV file
    for row in csv_reader:
        # Process the row data
        print(row)
```

3. For a reader object, the associated file object needs to be opened in read mode ('r') to allow the object to read data from the file. For a writer object, the associated file object needs to be opened in write mode ('w') to allow the object to write data to the file.

4. The writerow() method of a csv.writer object takes a list as its argument and writes it as a single row to the CSV file.

5.
   The delimiter keyword argument specifies the character that separates the fields in a CSV file. By default, the csv module assumes that fields are separated by commas (,), but you can specify a different delimiter using the delimiter keyword argument

6. The json.loads() function can be used to parse a JSON string and convert it into a Python data structure. The json.loads() function takes a string as its argument, and returns a corresponding Python data structure, such as a list, dictionary, or string, depending on the JSON data.

7. The json.dumps() function can be used to convert a Python data structure into a JSON string. The json.dumps() function takes a Python object as its argument and returns a corresponding JSON string.