In [1]:
```
# Q1 Excel is more superior when compared with the CSV file format; thus, CSV consumes less amount of file size
#    when the user is importing data; it is a much faster format compared to excel.
#     CSV does not manipulate data and stores it as-is. Excel also allows the user the add-in feature.
```

In [3]:
```
# Q2 To create a reader object using the csv.reader() function in Python's CSV module, you pass a file object that contains
#    the CSV data. For example, you can open a CSV file using the built-in open() function and pass the file object to csv.reader
#              To create a writer object using the csv.writer() function, you pass a file object that you want to write
#          the CSV data to. For example, you can create a file object using the built-in open() function and pass it to csv.write
```

In [4]:
```
# Q3 For a reader object created using csv.reader(), the file object should be opened in text mode with the newline parameter set
#    This is necessary because the CSV module handles newline characters itself.
```

In [5]:
```
# Q4 To write a list of data to a CSV file using the csv module in Python, you can use the writerow() method of a csv.
#    writer() object. This method takes a list argument representing a row of data and writes it to the CSV file.
```

In [6]:
```
# Q5 The delimiter argument specifies the character that separates values in each row of the CSV file. By default,
#     the delimiter is a comma (','). You can change it by passing a different character to the delimiter parameter
#     when creating a csv.writer()
```

In [7]:
```
# Q6 json.loads() function: This function takes a string of JSON data as input and returns a corresponding Python object.
#   the function name "loads" stands for "load string".
```

In [ ]:
```
# Q7 json.dump() function: This function writes a Python object to a file-like object in JSON format.
```