1. What advantages do Excel spreadsheets have over CSV spreadsheets?

Ans: Excel spreadsheets and CSV (Comma-Separated Values) spreadsheets are two popular formats for storing and analyzing tabular data. While both formats have their own advantages and disadvantages, Excel spreadsheets have several advantages over CSV spreadsheets:

Formatting: Excel spreadsheets allow for more complex formatting options, such as cell borders, colors, and font styles. This can be useful when presenting data to others or when you want to make the data easier to read and understand.

Formulas and functions: Excel allows you to use formulas and functions to perform calculations and manipulate data. This can be particularly useful when working with large datasets that require complex calculations or when you need to perform the same calculation across multiple cells.

Data validation: Excel allows you to set up data validation rules to ensure that data is entered correctly and conforms to a certain format or set of values. This can help to prevent errors and inconsistencies in the data.

Sorting and filtering: Excel allows you to easily sort and filter data based on specific criteria, making it easier to analyze and work with large datasets.

Data visualization: Excel provides a range of charting and graphing options, which can be used to visualize data in different ways and gain insights into trends and patterns.

2.What do you pass to csv.reader() and csv.writer() to create reader and writer objects?

In [ ]:
```
import csv
with open('text.csv','r') as file:
    csv_file = csv.reader(file,delimiter=',')
    for ele in csv_file:
        print(ele)
```

3. What modes do File objects for reader and writer objects need to be opened in?

Ans: For `csv.reader(iterable_file_object)` , the file objects needed to be opened in `read` mode `mode='r'` Whereas for `csv.writer(iterable_file_object)` the file objects needed to be opened in `write` mode `mode='w'`

4. What method takes a list argument and writes it to a CSV file?

Ans: csv.writer class provides two methods for writing to CSV. They are `writerow()` and `writerows().` writerow() method writes a single row at a time. Whereas writerows() method is used to write multiple rows at a time.

In [2]:
```
# Example Program
import csv
fields = ['Name', 'Branch', 'Year', 'CGPA'] #column names
rows = [
            ['Nikhil', 'COE', '2', '9.0'],  # data rows of csv file
            ['Sanchit', 'COE', '2', '9.1'],
            ['Ravi', 'IT', '2', '9.3']
       ]
with open("university_records.csv", 'w') as csvfile:
    csvwriter = csv.writer(csvfile) # creating a csv writer object
    csvwriter.writerow(fields) # writing the fields
    csvwriter.writerows(rows) # writing the data rows
```

5. What do the keyword arguments delimiter and line terminator do?

Ans: Lets take the example of a csv file:
```
First Name, Last Name, Age
shri, krishna, 26
krishna, ram, 27
```
Here `','` is Delimiter. We can use any Character as per our needs if required. Similarly Line Terminator comes at end of line by default it is newline and can be changed accourding to Requirement.

6. What function takes a string of JSON data and returns a Python data structure?

Ans: `loads()` method takes a string of JSON data and returns a Python data structure

In [3]:
```
# Example of json.loads() method
import json
my_details_json ='''{
    "Name": "Mano Vishnu",
    "Qualification": "Bachelor of Technology",
    "Stream": "Computer Science and Engineering"
}'''
print(my_details_json)
print(f'Type of my_details_json is {type(my_details_json)}')
my_details = json.loads(my_details_json)
print(my_details)
print(f'Type of my_details is {type(my_details)}')
```

```
{
    "Name": "Mano Vishnu",
    "Qualification": "Bachelor of Technology",
    "Stream": "Computer Science and Engineering"
}
Type of my_details_json is <class 'str'>
{'Name': 'Mano Vishnu', 'Qualification': 'Bachelor of Technology', 'Stream':
'Computer Science and Engineering'}
Type of my_details is <class 'dict'>
```

7. What function takes a Python data structure and returns a string of JSON data?

Ans: `dumps()` method takes a python data structure and returns a string of JSON data

In [4]:
```
# Example of json.dumps() method
import json
my_details = {
    'Name':'Mano Vishnu',
    'Stream':'Computer Science and Engineering',
    'Qualification':'Bachelor of Technology'
}
print(my_details)
print(f'Type of my_details is {type(my_details)}')
my_details_json = json.dumps(my_details, indent=4, sort_keys=True)
print(my_details_json)
print(f'Type of my_details_json is {type(my_details_json)}')
```

```
{'Name': 'Mano Vishnu', 'Stream': 'Computer Science and Engineering', 'Qualif
ication': 'Bachelor of Technology'}
Type of my_details is <class 'dict'>
{
    "Name": "Mano Vishnu",
    "Qualification": "Bachelor of Technology",
    "Stream": "Computer Science and Engineering"
}
Type of my_details_json is <class 'str'>
```