Json

- Json stands for JavaScript Object Notation. It is a lightweight format that is used for storing and transporting data.

- It has a self describing in nature as it uses **human-readable** text to store and transmit data objects consisting of **key-value pairs**.

- It is a language independent data format and was derived from javascript

- It is commonly used for transmitting data in web applications (e.g., sending some data from the server to the client, so it can be displayed on a web page, or vice versa).

Advantages of Json:

1. JSON is light weight and Faster

2. It has a self describing nature

3. It supports nested properties which helps to share large and complex data easily.

4. It can easily be validated by json schema

It follows the following syntax rules:

- The data is always in key/value pairs.
- A comma separates every data object.
- The objects are held within the curly braces '{ }'.
- The arrays are held in square brackets.

Datatypes

Values must have one of the following data types.

1. Strings:

- A double-quoted Unicode that contains characters with backslash escaping
- Eg **:** {"name": "Abc"}

2. Number:

- Used for any numeric type, either integers or floating point numbers.
- Eg **:** {"Amount" : 21.5}

3. Integer

- Used for integral values

- Eg : {“Amount” : 21}

4. Object:

- It is a set of keys or value pairs inserted between { } (curly braces). The keys must be strings and should be unique and multiple keys and value pairs are separated by a, (comma).

- Eg **:** {“ Address ” : {“ Country ”: “ India ”, “ State ”: “ Gujarat“}}

5. Boolean:

- This data type can be either true or false.
- Eg **:** {“sale”: true}

6. Arrays:

- It is an ordered collection of values inserted between [ ] (Square brackets). The values of array are separated by , (comma).

- Eg: [value, …….]

- **Here is a small example of a random generated json file of 10,000 people performing transactions that represents**

1. UID (User Identification Number)
2. Name
3. DOB (Date Of Birth)
4. Gender
5. Address (place where the transaction occurred)

a. Country

b. State

6. Account No (Account Number of the person)
7. To Account No (account number of the person to which payment is done)
8. Timestamp (date and time at which transaction has occurred)
9. Status (transaction is successful or not)

```
[{
        "UID": "T1",
        "Name": "Jennifer Castillo",
        "DOB": "10-01-94",
        "Gender": "F",
        "Address": {
            "Country": "India",
            "State": "Andhra Pradesh"
        },
        "Account No": 4000000001,
        "To Account No": 5000000001,
        "Transaction Amount": 25.0,
        "Timestamp": "02-08-16 14:32:07:000000",
        "Status": "success"
    },
    {
        "UID": "T2",
        "Name": "Sophia Calderon",
        "DOB": "04-04-57",
        "Gender": "M",
        "Address": {
            "Country": "India",
            "State": "Arunachal Pradesh "
        },
        "Account No": 4000000002,
        "To Account No": 5000000002,
        "Transaction Amount": 27999.0,
        "Timestamp": "02-08-16 14:18:58:000000",
        "Status": "success"
    },
    {
        "UID": "T3",
        "Name": "Brian Wallace",
        "DOB": "26-11-96",
        "Gender": "F",
        "Address": {
            "Country": "India",
            "State": "Assam"
        },
        "Account No": 4000000003,
        "To Account No": 5000000003,
        "Transaction Amount": 459.0,
        "Timestamp": "02-08-16 14:27:12:000000",
        "Status": "success"
    }]
```

Json Schema

- JSON Schema is a vocabulary that helps in validating JSON documents.

- It describes your existing data format and Provides a clear human- and machine- readable documentation.

- We can add properties or constraints to it, which makes the json file more specific and simpler for further analysis. We can add:
  - Data types of elements
  - Formats for values of the elements
  - Restrictions on the length of the values
  - Enumeration for an element with a fixed set of values etc

- Like, "Type" is one of the most widely used keywords which checks that the type of the JSON value is one of a few specified types (object, array, string, number, boolean, or null)

- Syntax) name : { "type" : "string" }

- This means that any other data type other than string will cause a validation error when checked against this schema; only strings will pass the validation.

- Here is an example of a json schema of the above json file, which have the following properties : -

```
{
    "title": "Transactions",
    "description": "A random generated file",

    "type": "array",
    "items" :
    {
        "type" : "object",
        "properties":
        {
            "UID" :
            {
                "type" : "string",
                "minLength" : 2,
                "maxLength" : 7,
                "pattern": "^T"
            },

            "Name" :
            {
                "type" : "string",
                "minLength" : 2,
                "maxLength" : 30
            },

            "DOB" :
            {
                "type" : "string",
                "pattern": "^[0-9][0-9]-[0-9][0-9]-[0-9][0-9]$"
            },

            "Gender" :
            {
                "type" : "string",
                "pattern": "^M|F$"
            },

            "Address" :
            {
                "type" : "object",
                "properties":
                {
                    "Country" :
                    {
                        "type" : "string",
                        "pattern" : "^India$"
                    },
                    "State" :
                    {
                        "type" : "string",
                        "enum": [


"Andhra Pradesh","Arunachal Pradesh
","Assam","Bihar","Chhattisgarh","Goa","Gujarat","Haryana",
                        "Himachal Pradesh","Jammu and
Kashmir","Jharkhand","Karnataka","Kerala","Madhya
Pradesh","Maharashtra",

"Manipur","Meghalaya","Mizoram","Nagaland","Odisha","Punjab","Raja
sthan","Sikkim","Tamil Nadu","Telangana",
                        "Tripura","Uttar
Pradesh","Uttarakhand","West Bengal","Andaman and Nicobar
Islands","Chandigarh",
                        "Dadra and Nagar Haveli","Daman and
Diu","Lakshadweep","National Capital Territory of
Delhi”,"Puducherry"
]
                    }
                },

                "required" : ["Country","State"],
                "additionalProperties" : false
            },

            "Account No" :
            {
                "$ref": "#/$defs/acc"
            },

            "To Account No" :
            {
                "$ref": "#/$defs/to_acc"
            },

            "Transaction Amount" :
            {
                "type" : "number",
                "minimum" : 0
            },

            "Timestamp" :
            {
                "type" : "string",
                "pattern": "^[0-9][0-9]-[0-9][0-9]-[0-9][0-9]
[0-9][0-9]:[0-9][0-9]:[0-9][0-9]:[0-9][0-9][0-9][0-9][0-9][0-9]$"
            },
            "Status" :
            {
                "type" : "string",
                "pattern" : "^success|failure$"
            }

        },

        "required" : [
"UID","Name","DOB","Gender","Address","Account No","To Account
No","Transaction Amount”,"Timestamp","Status"
],
        "additionalProperties" : false
    },


    "$defs" :
    {
        "acc" :
        {
            "type" : "integer",
            "minimum" : 4000000000,
            "exclusiveMaximum" : 5000000000
        },
        "to_acc" :
        {
            "type" : "integer",
            "minimum" : 5000000000,
            "exclusiveMaximum" : 6000000000
        }
    },

    "examples" :
    [
        {
            "UID": "T1",
            "Name": "Jennifer Castillo",
            "DOB": "10-01-94",
            "Gender": "F",
            "Address": {
                "Country": "India",
                "State": "Andhra Pradesh"
            },
            "Account No": 40000000001,
            "To Account No": 50000000001,
            "Transaction Amount": "25",
            "Timestamp": "02-08-16 14:32:07:000000",
            "Status": "success"
        }
    ]
}
```

- **Properties which I have used in the schema is :**

1. **Type** (Tells about the data type of the key)

2. **MinLength** (used to define minimum length of the string)

3. **MaxLength** (used to define maximum length of the string)

4. **Pattern (**

5. **Enum(Enumeration)**

- To restrict a value to a fixed set of values, **enum** keyword can be used and it must be an array with at least one element
- Each element of the enum is unique

6. **Required**(By default, the properties are optional. The list of mandatory properties can be provided using the keyword, required.)

7. **Additional properties**(To restrict the caller from providing additional properties with the data that are not defined in the schema, the keyword additionalProperties can be used)

8. **DEF**

9. **REF**

Validation Of Json

- Json can be validated by json schema using python as it has an inbuilt structure called dictionary which similar to json.

- The JSON document which is being validated is called an **Instance**, and the document containing the description is called a **Schema** which contains list of rules that is collected into a JSON object.

- **Steps to validate,**

  1. Jsonschema.validate is an inbuilt function which validates json file with the corresponding schema

  2. Load the json file and the schema with the help of json.load and store it under a variable name

  3. Add the json file and the schema in the validation function as parameters

Uploading To Database

- After validation of json, data is uploaded to the database for further analysis.

- A CSV file is generated of the instance, that does not match with the json schema.