**CSV Format**

Eg:

id, name, email,age,

1, Alice, alice@example.com, 25, 100.0, 2024-01-10,

2, Bob, bob@example.com, None, 500.5, 01-12-2024

3, Charlie, charlie@domain, 30, None, 2024/02/15

4, David, None, 28, 300.0, 05/03/2024

5, Eve, eve@example.com, Twenty-Two, 700.75, 2024-01-25

1, Alice, alice@example.com, 25, 100.0, 2024-01-10

7, John, JOHN@GMAIL.COM , 40, 50.5, Feb 20, 2024

8, Mike, mike@example.com, 35, NA, 2024-01-30

– comma separated values

– Human readable format

– everything in string type

– each row is a record

– doesn’t support complex datatypes

– inefficient for large datasets

Reading csv file:

df = spark.read.csv('dbfs:/FileStore/source\_to\_bronze/employee\_df.csv', header=True)

**JSON Format**

Eg:

[

[ {

"id": 1,

"name": "John Doe",

"age": 30,

"email": "john.doe@example.com",

"address": {

"street": "123 Elm St",

"city": "Springfield",

"zip": "12345"

},

"orders": [

{"order\_id": 101, "product": "Laptop", "amount": 1200.50, "date": "2022-01-15"},

{"order\_id": 102, "product": "Smartphone", "amount": 800.99, "date": "2022-03-22"}

]

},

{

"id": 2,

"name": "Jane Smith",

"age": 27,

"email": "jane.smith@example.com",

"address": {

"street": "456 Oak St",

"city": "Metropolis",

"zip": "67890"

},

"orders": [

{"order\_id": 103, "product": "Tablet", "amount": 499.99, "date": "2022-04-10"}

]

}]

– A text based format where everything stored as a key-value pairs

– supports nested structure

– commonly used for API’s, configurations, transfer data through web

Reading JSON File:

df = spark.read.json('dbfs:/FileStore/sample\_json\_3\_2.json', multiLine=True)

**Parquet Format**

– A columnar storage format

– it is not Human readable because everything in Binary format

– faster performance when reading columns

– compressed for large datasets, optimized for better performance

– more complex than csv, json

– required special libraries, tools to read it

Reading Parquet File:

df = spark.read.parquet('dbfs:/FileStore/shared\_uploads/arjun37ca@gmail.com/mtcars.parquet', inferSchema=True)

**Delta Format**

– An extension of parquet, added features like ACID properties, versioning of datalakes

– built on top of apache spark

– it is also a binary format,but includes metadata like transaction logs

– supports incremental data processing , schema enforcement, time travel

Reading Delta files:

df = spark.read.format('delta').load('dbfs:/Users/arjun37ca@gmail.com/d1')

**YAML File**

eg:- name: Alice

age: 30

city: New York

– A human readable data serialization format, where data stored with indentation.

– used in configuration files(docker, kubernetes)

– easy to write by humans and supports nested complex structures.

Reading YAML File:

import yaml

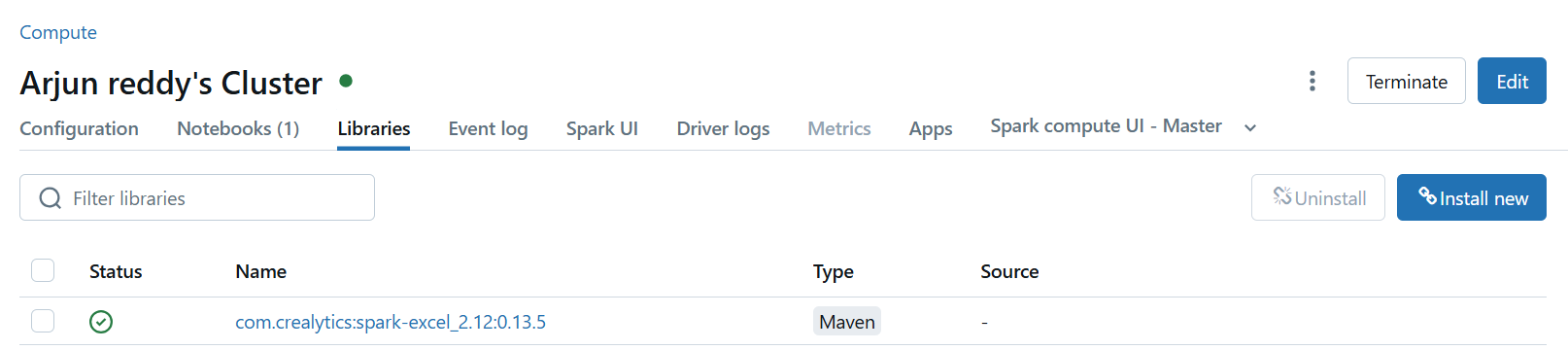
with open("config.yaml", "r") as file:

data = yaml.safe\_load(file)

print(data)

**Reading xlsx file:**

Select “**Maven**” as the Library source. In the “**Coordinates**” field, copy and paste the following: “**com.crealytics:spark-excel\_2.12:0.13.5**”.



excel\_file\_path = "dbfs:/FileStore/tables/pokemon\_data.xlsx"

df = spark.read \

.format("com.crealytics.spark.excel") \

.option("header", "true") \

.option("inferSchema", "true") \

.load(excel\_file\_path)