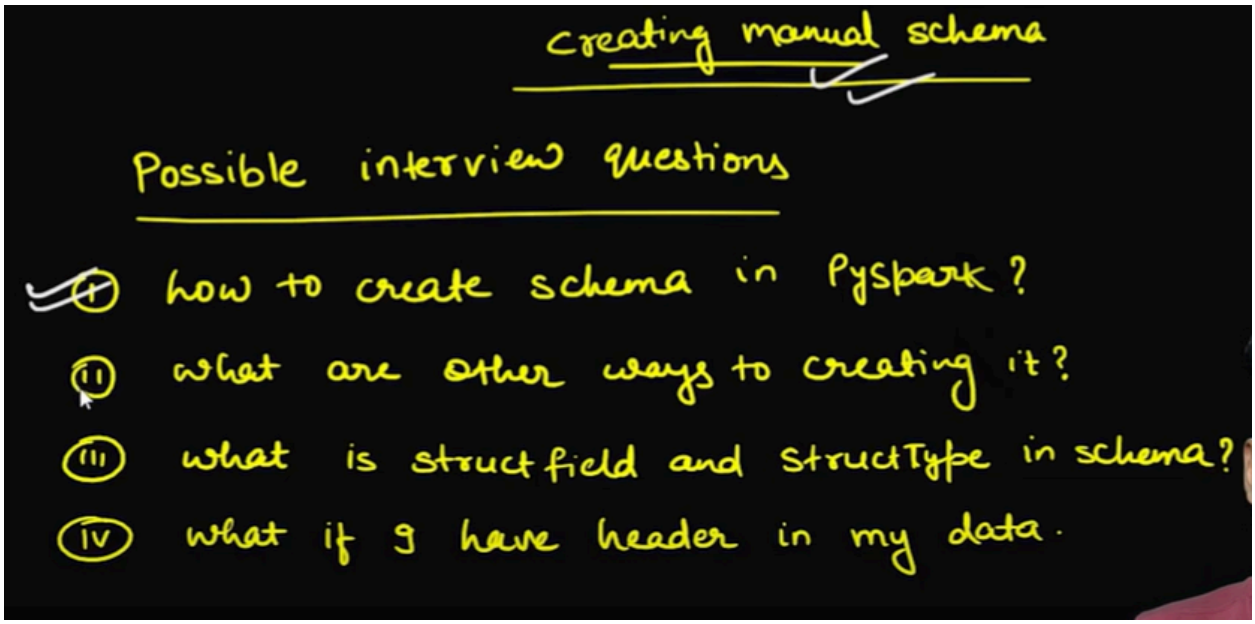
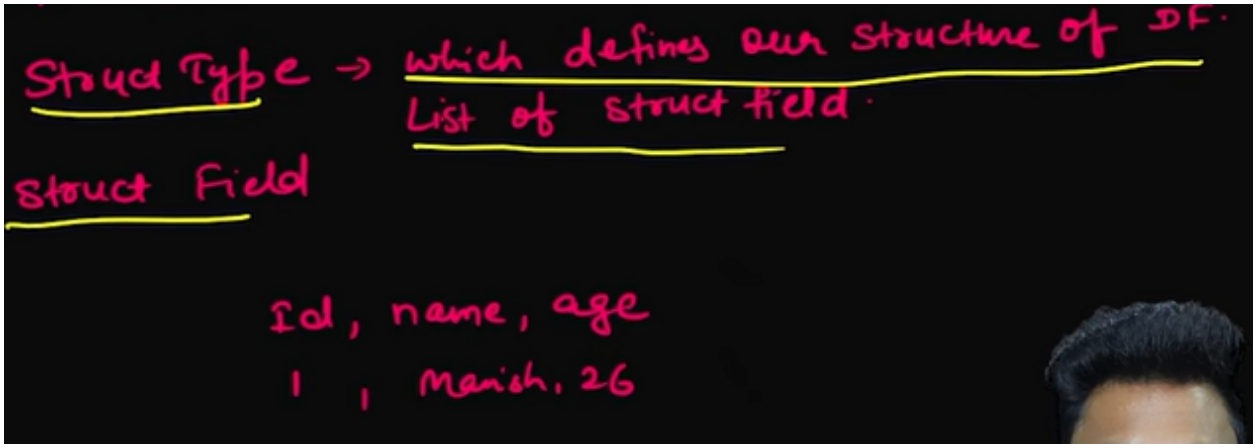
Schema in spark

1. 

2. Coming to first question
3. We can define schema in two ways


4. To use struct type and field ..we have to import them from pyspark.sql
5. Lets consider a sample data

mId	name	age
1	Manish	26
2	Vikash	30
3	Pritam	22

6. 

7. Example in code

```
customer_schema = StructType([
  StructField("name", StringType(), True),
  StructField("age", IntegerType(), True),
  StructField("purchases", ArrayType(StringType()), True)
])

# Create an empty DataFrame with the defined schema
customers_df = spark.createDataFrame([], customer_schema)
```

8. Defining Schema using DDL
9. Example :

```
# Example DDL string
ddl_string = "`name` STRING, `age` INT, `active` BOOLEAN"
```

10. Practical..imported all the required functions and implemented my_schema

Cmd 6

```python
from pyspark.sql.types import StructField,StructType,StringType,
IntegerType
```

Command took 0.04 seconds -- by kaushikvarma958@gmail.com at 3/9/2024, 4:52:35 PM on My Cluster

Cmd 7

Python

```python
my_schema = StructType([
    StructField("DEST_COUNTRY_NAME",StringType(),True),
    StructField("ORIGIN_COUNTRY_NAME",StringType(),True),
    StructField("Count",IntegerType(),True),

])
```

11. Here when we run below code..we are facing with error

Python

```python
flight_df = spark.read.format("csv")\
    .option("header","false")\
    .option("inferschema","false")\
    .schema(my_schema)\
    .option("mode","FAILFAST")\
    .load("/FileStore/tables/Flight_data_2010.csv")
flight_df.show()

```

(1) Spark Jobs

```
org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 16.0 failed 1 times, most recent failure: Lost task 0.0 in stage 16.0 (TID 16) (ip-10-172-209-115.us-west-2.compute.internal executor driver): com.databricks.sql.io.FileReadException: Error while reading file dbfs:/FileStore/tables/Flight_data_2010.csv.
```

Command took 2.24 seconds -- by kaushikvarma958@gmail.com at 3/9/2024, 4:57:30 PM on My Cluster

12. That is because of FAILFAST mode..

13. Here we have a null value in the count

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME| null|
|    United States|       Saint Martin|    2|
```

and we have defined count column as non nullable..so failfast mode is treating this as malformed

14. We have defined myschema which includes the header and when reading a file..if we choose .option("header",false) then it gives result like this..because dataframe api is reading header from myschema

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME| null|
```

15. To avoid that we use ".option("skipRows",1)..it skips the first row in the file

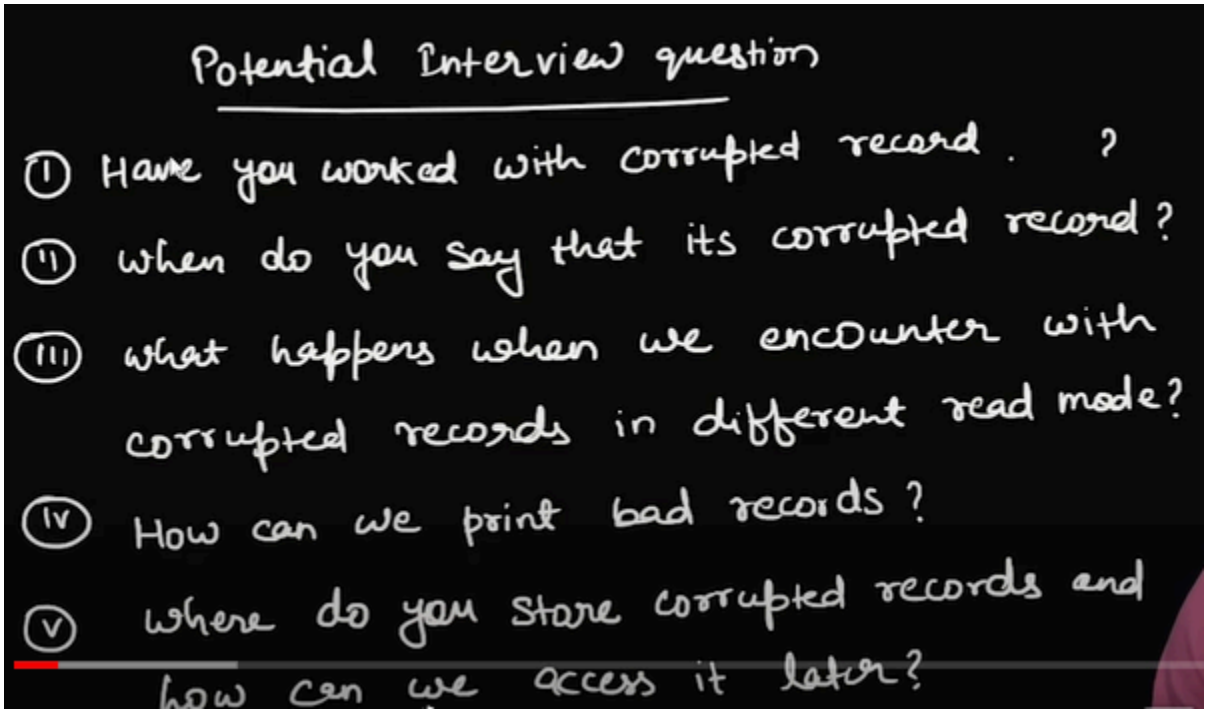
```
flight_df=spark.read.format("csv")\
            .option("header","false")\
            .option("skipRows",1)\
            .option("inferschema","false")\
            .schema(my_schema)\
            .option("mode","PERMISSIVE")\
            .load("/FileStore/tables/flight_data.csv")

flight_df.show(5)
```

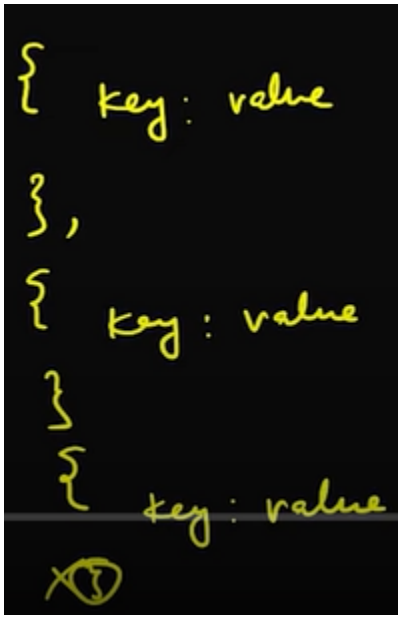
16. So in the end ..we have answered this four questions in the schema

Handling Corrupted records in Spark

1. Potential interview questions while handling corrupted data in spark



2. First we'll start with 2nd question
3. Lets take two file formats..CSV and JSON
4. In JSON we have curly brackets and inside them we have a key:value pair



5. Here in the above image ..we can see that there is no closing bracket for 3rd key-value pair. So we can say that it is a corrupted record
6. Lets look at corrupted record in CSV

7. Here we have a sample csv file

```
id,name,age,salary,address,nominee
1 , Manish , 26 , 75000 , bihar , nominee1
2 , Nikita , 23 , 100000 , uttarpradesh , nominee2
3 , Pritam , 22 , 150000 , Bangalore,India , nominee3
4 , Prantosh , 17 , 200000 , Kolkata,India , nominee4
5 , Vikash , 31 , 300000 , , nominee5
```

8. Here for 3rd record..we have bangalore,India…which our schema is not defined to handle two values for address..so it is one corrupted record
9. Here 5th row is not corrupted
10. Now lets answer this questions

How many records will be there in permissive, drop malformed and failfast respectively ?

11. In permissive we get all rows(makes corrupted values as null)..in DropMalformed we get 3 records..and failfast will give an error .if there are any corrupted values

12. So to store corrupted records and to print them

- `PERMISSIVE`: when it meets a corrupted record, puts the malformed string into a field configured by `columnNameOfCorruptRecord`, and sets malformed fields to `null`. To keep corrupt records, an user can set a string type field named `columnNameOfCorruptRecord` in an user-defined schema. If a schema does not have the field, it drops corrupt records during parsing. A record with less/more tokens than schema is not a corrupted record to CSV. When it meets a record having fewer tokens than the length of the schema, sets `null` to extra fields. When the record has more tokens than the length of the schema, it drops extra tokens.

13. Practicals

14. We have created a corrupted csv file

```
corrupted_df = spark.read.format("CSV")\
    .option("header","true")\
    .option("inferSchema","true")\
    .option("mode","PERMISSIVE")\
    .load("/FileStore/tables/Corruped-1.csv")
corrupted_df.show()
```

▸ (3) Spark Jobs

▸ corrupted_df: pyspark.sql.dataframe.DataFrame = [id: intege
fields]

```
+---+--------+---+------+------------+--------+
| id|    name|age|salary|     address| nominee|
+---+--------+---+------+------------+--------+
|  1|  Manish| 26| 75000|       bihar|nominee1|
|  2|  Nikita| 23|100000|uttarpradesh|nominee2|
|  3|  Pritam| 22|150000|   Bangalore|   India|
|  4|Prantosh| 17|200000|     Kolkata|   India|
|  5|  Vikash| 31|300000|        null|nominee5|
+---+--------+---+------+------------+--------+
```

15. Here we can see permissive mode has ignored the values of nominee column for row3,4
16. And for it gives null values..if the value is not present
17. Same data in MALFORMED

18.
```python
corrupted_df = spark.read.format("CSV")\
    .option("header","true")\
    .option("inferSchema","true")\
    .option("mode","DROPMALFORMED")\
    .load("/FileStore/tables/Corruped-1.csv")
corrupted_df.show()
```

```
▸ (3) Spark Jobs

▸ corrupted_df: pyspark.sql.dataframe.DataFrame = [id: integ
fields]

+---+------+---+------+------------+--------+
| id|  name|age|salary|     address| nominee|
+---+------+---+------+------------+--------+
|  1|Manish| 26| 75000|       bihar|nominee1|
|  2|Nikita| 23|100000|uttarpradesh|nominee2|
|  5|Vikash| 31|300000|        null|nominee5|
```

19. As we can see drop malformed has deleted the corrupted records rows(3,4) and it gives null value if there is no value
20. FAILFAST
21. Failfast gives an error

```python
employee_df=spark.read.format("csv")\
        .option("header","true")\
        .option("inferschema","true")\
        .option("mode","FAILFAST")\
        .load("/FileStore/tables/employee_file.csv")
employee_df.show()
```

```
▸ (3) Spark Jobs
org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in
ent failure: Lost task 0.0 in stage 8.0 (TID 8) (ip-10-172-248-128.us-west-2
com.databricks.sql.io.FileReadException: Error while reading file dbfs:/File
Command took 3.55 seconds -- by manisnitt@gmail.com at 3/29/2023, 8:57:45 AM on My Cluster
```

22. How can we print corrupted records?

23. First we have define “*corruped*_record” in the schema

```
emp_schema= StructType(
                        [
                            StructField("id",IntegerType(),True),
                            StructField("name",StringType(),True),
                            StructField("age",IntegerType(),True),
                            StructField("salary",IntegerType(),True),
                            StructField("address",StringType(),True),
                            StructField("nominee",StringType(),True),
                            StructField("_corrupt_record", StringType(), Tru
```

24. Next we will pass our schema ..while reading

```
Cmd 18

employee_df=spark.read.format("csv")\
        .option("header","true")\
        .option("inferschema","true")\
        .option("mode","PERMISSIVE")\
        .schema(emp_schema)
        .load("/FileStore/tables/employee_file.csv")
employee_df.show()
```

25. The output

```
▸ (1) Spark Jobs
▸ employee_df: pyspark.sql.dataframe.DataFrame = [id: integer, name: string ... 5 more fields]

+---+--------+---+------+------------+--------+--------------------+
| id|    name|age|salary|     address| nominee|     _corrupt_record|
+---+--------+---+------+------------+--------+--------------------+
|  1|  Manish| 26| 75000|       bihar|nominee1|                null|
|  2|  Nikita| 23|100000|uttarpradesh|nominee2|                null|
|  3|  Pritam| 22|150000|   Bangalore|   India|3,Pritam,22,15000...|
|  4|Prantosh| 17|200000|     Kolkata|   India|4,Prantosh,17,200...|
|  5|  Vikash| 31|300000|        null|nominee5|                null|
+---+--------+---+------+------------+--------+--------------------+

Command took 1.82 seconds -- by manisnitt@gmail.com at 3/29/2023, 9:01:01 AM on My Cluster
Cmd 19
```

26. Here from the output ..we can see there are corrupted rows in the corrupted_record column
27. To get the full details in corrupt_record columns..we use truncate = true

```
        .load("/FileStore/tables/employee_file.csv")
employee_df.show(truncate = False)
```

▸ (1) Spark Jobs

▸ employee_df: pyspark.sql.dataframe.DataFrame = [id: integer, name: string ... 5 more fields]

```
+---+--------+---+------+------------+--------+-----------------------------------------
|id |name    |age|salary|address     |nominee |_corrupt_record
+---+--------+---+------+------------+--------+-----------------------------------------
|1  |Manish  |26 |75000 |bihar       |nominee1|null
|2  |Nikita  |23 |100000|uttarpradesh|nominee2|null
|3  |Pritam  |22 |150000|Bangalore   |India   |3,Pritam,22,150000,Bangalore,India,nominee
|4  |Prantosh|17 |200000|Kolkata     |India   |4,Prantosh,17,200000,Kolkata,India,nominee
|5  |Vikash  |31 |300000|null        |nominee5|null
+---+--------+---+------+------------+--------+-----------------------------------------
```

28.
29. How to store corrupted recorded?
30. SO while reading the data ..we have to pass an option which has key and value(path)..here we should not option “MODE” while storing bad_records

```
employee_df=spark.read.format("csv")\
        .option("header","true")\
        .option("inferschema","true")\
        .option("mode","PERMISSIVE")\
        .schema(emp_schema)\
        .option("badRecordsPath","/FileStore/tables/bad_recods")
        .load("/FileStore/tables/employee_file.csv")
employee_df.show(truncate = False)
```

▸ (1) Spark Jobs

▸ employee_df: pyspark.sql.dataframe.DataFrame = [id: integer, name: string ... 5 more fields]

```
+---+------+---+------+------------+--------+---------------+
|id |name  |age|salary|address     |nominee |_corrupt_record|
+---+------+---+------+------------+--------+---------------+
|1  |Manish|26 |75000 |bihar       |nominee1|null           |
|2  |Nikita|23 |100000|uttarpradesh|nominee2|null           |
|5  |Vikash|31 |300000|null        |nominee5|null           |
+---+------+---+------+------------+--------+---------------+
```

31. Output :
32. To verify the files in our file system ..we use

```
%fs
ls /FileStore/tables/
```

33.

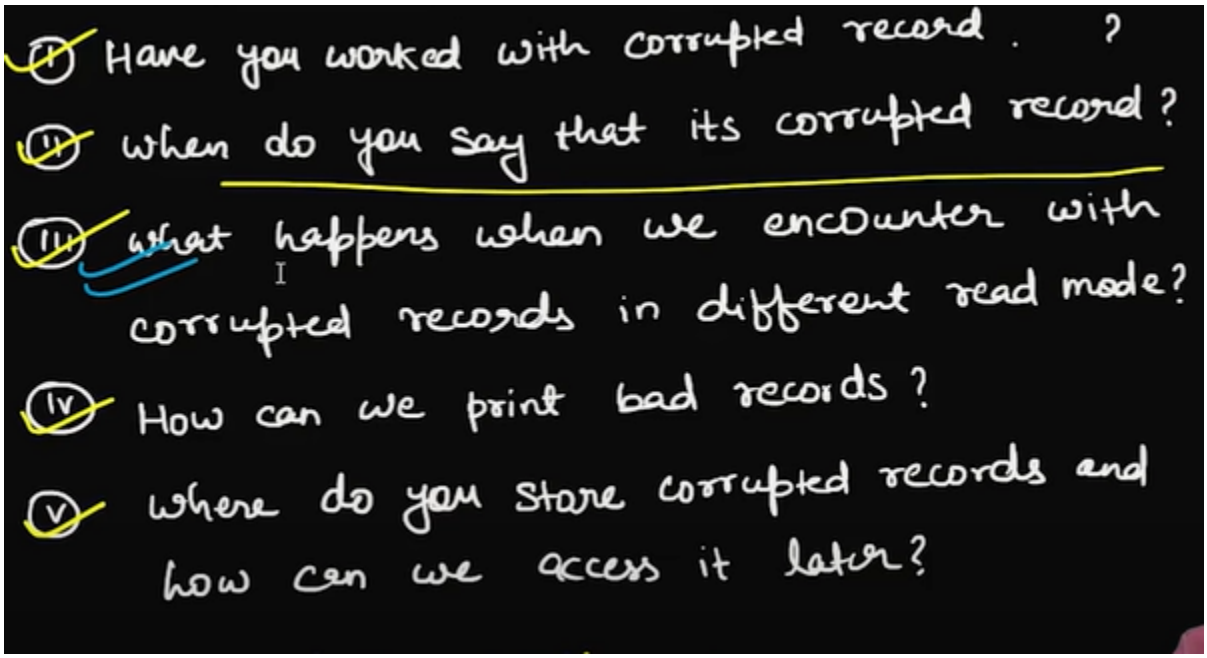
	path	name	size	modificationTime
1	dbfs:/FileStore/tables/bad_recods/	bad_recods/	0	0
2	dbfs:/FileStore/tables/employee_file.csv	employee_file.csv	230	1680060088000
3	dbfs:/FileStore/tables/flight_data.csv	flight_data.csv	7323	1679805900000

34. Here we can see..it created a file for bad_records
35. By default this files will get stored in JSON format..so to access them we use

```
bad_data_df= spark.read.format("json").load("/FileStore/tables/bad_recods/20230329T033518/bad_records/")
bad_data_df.show()
```

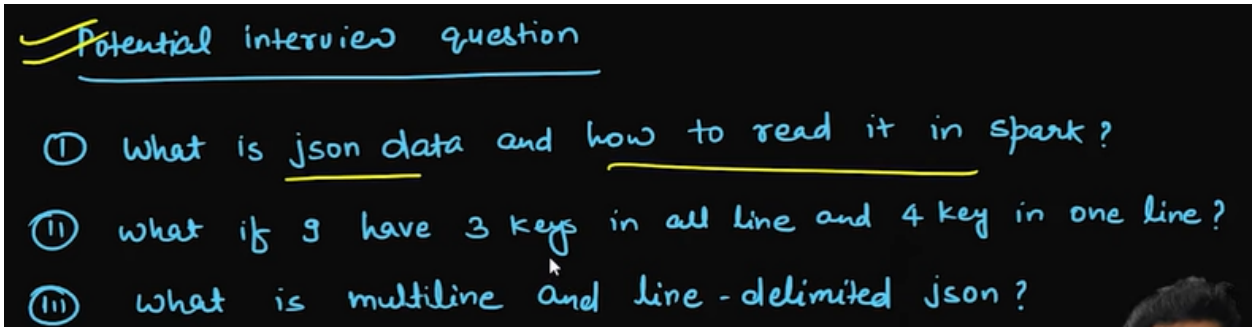
```
▸ (2) Spark Jobs
▸ bad_data_df: pyspark.sql.dataframe.DataFrame = [path: string, reason: string ... 1 more field]
+--------------------+--------------------+--------------------+
|                path|              reason|              record|
+--------------------+--------------------+--------------------+
|dbfs:/FileStore/t...|org.apache.spark....|3,Pritam,22,15000...|
|dbfs:/FileStore/t...|org.apache.spark....|4,Prantosh,17,200...|
+--------------------+--------------------+--------------------+
```

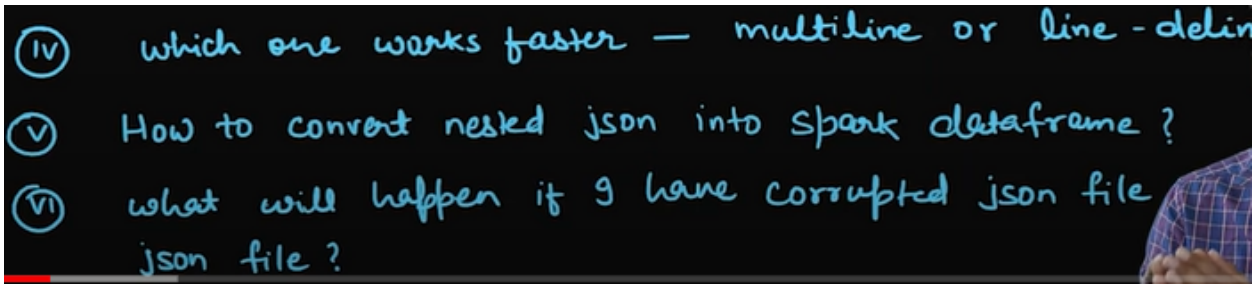
36. So in end we have answered



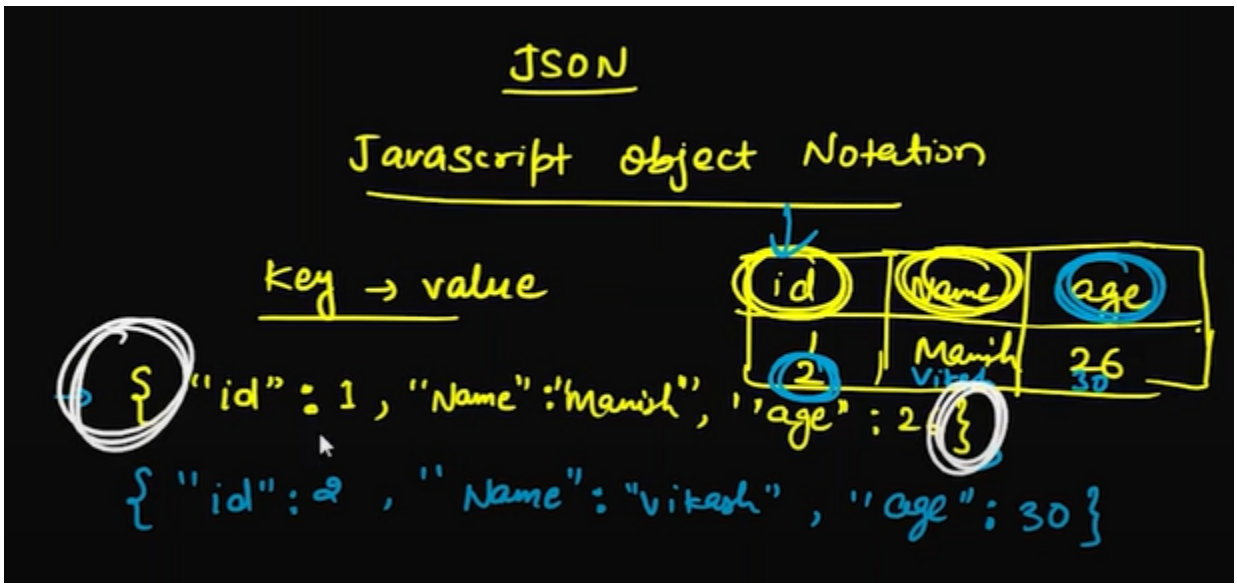
How to read JSON files

1. Potential interview question

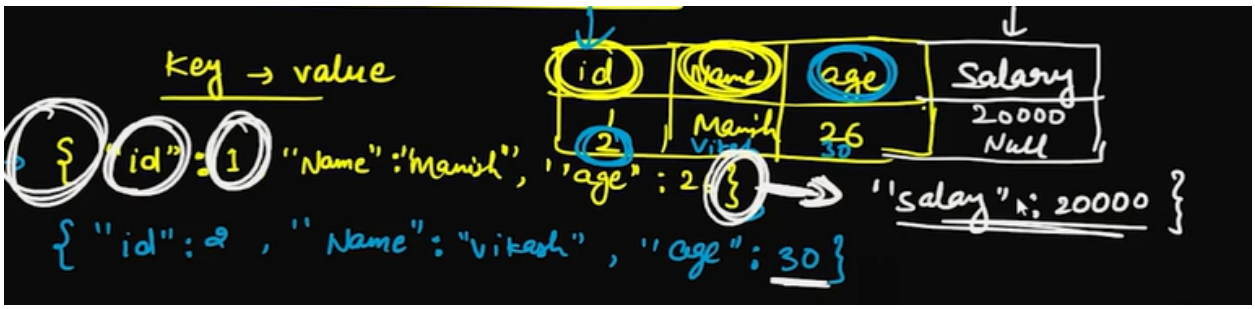




2. 6th Ques : json file or invalid json file?
3. 5th question is very imp in real time
4. Json is a key-value pair..the way is stores is below



5. Here JSON is a semi structured data..for example we can add any values to id 1..with out adding them in id 2…but in structured data like csv..we have to add values to every row



6. How to read json data in spark

7. Here I have uploaded multiple types of json files

```
File uploaded to /FileStore/tables/corrupted.json
File uploaded to /FileStore/tables/employe-1.json
File uploaded to /FileStore/tables/Multi_line_correct.json
File uploaded to /FileStore/tables/Multi_line_incorrect.json
File uploaded to /FileStore/tables/single_file_json_extra_fields.
json
```

8. First we will use line_delimited_json

```
{"name":"Manish","age":20,"salary":20000},
{"name":"Nikita","age":25,"salary":21000},
{"name":"Pritam","age":16,"salary":22000},
{"name":"Prantosh","age":35,"salary":25000},
{"name":"Vikash","age":67,"salary":40000}
```

9. Refer : https://www.youtube.com/watch?v=M0Kx205dxmM&list=PLTsNSGeIpGnGjaMSYVlidqVWSjKWoBhbr&index=7
10. Line delimited vs multi line : https://g.co/gemini/share/0019d1dfc26f
11.