**Week 3 day 2 (19-11-2024)**

**Topics covered:**

1. **Various features and working of spark in Databricks**
2. **About clusters**
3. **Coding:**

|  |  |  |
| --- | --- | --- |
| **S.no.** | **Topic** | **Code Reference** |
| **1.** | Importing, Creating, initialising a spark | refer Pyspark programs.ipynb |
| **2.** | Create RDD |  |
| **3.** | Data frames from rdd |  |
| **4.** | creating table, adding data etc. |  |
| **5.** | working on datasets:  (functions in Spark)  Importing csv file, head, column, drop, fill, rename, add, show, filter, groupby, agg, etc. | Refer  Working\_on\_Datasets.ipynb |

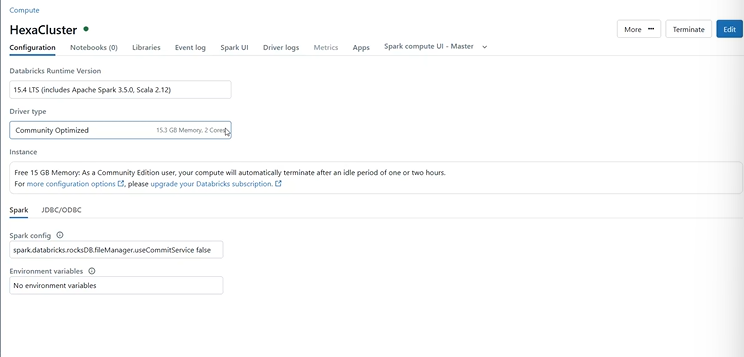
**Data bricks:**

Databricks is a data warehouse which provide data bricks dashboard for us, where we are going to create a cluster, in which our spark will run.

Eg. Here we are creating a cluster with 15 GB memory.

**Spark Concepts:**

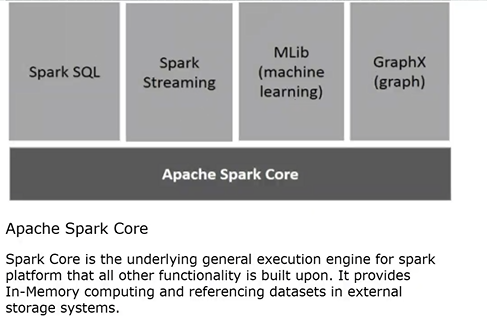
**(Cluster created image from day 1)**



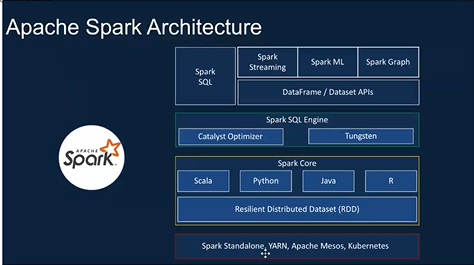
Spark uses Hadoop only for

Spark core is a main analytical engine which will have API(Py Spark) (languages)

Spark also have 4 modules: spark sql, spark streaming, MLlib, GraphX.



In Apache.org entire package downloads.



**Languages available? or**

**How many set of dataset API in spark?**

1. Scala
2. Python
3. java
4. R

**Modules available?**

1. spark sql,
2. spark streaming,
3. MLlib,
4. GraphX and so on.

**Spark Set up methods:**

That last block in spark architecture.

1. Spark stand alone,
2. Yarn,
3. Apache Mesos,
4. Kubernetes.

In general words we will tell like:

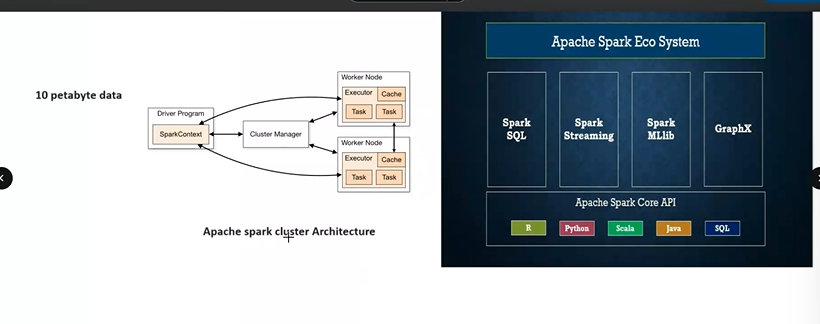
Spark

Locally

Databricks

Kubernetes (open source tool)

In yarn (stand alone manager)



**How we setup data cluster in data bricks?**

As we did in hands-on

**Cluster:**

Similar to aws instance. Cluster is a machine with certain GB memory. (eg our 15 GB cluster) which will be created in cloud remotely.

A cluster is a node. (means we have created 25 GB

S1: Spark context

Every cluster have a manager, executor, cache

**Cluster feature:**

Mostly cluster will have inbuild spark.

So, whenever we try to run a program it will approach inbuild Spark inside cluster as we are linking our notebook with cluster.

If this inbuild spark is not there then, we have to install this spark in our local machine. And this (15 GB + 2 core) cluster will start execute in our local PC.

Inside the spark we have spark core. Spark core will help us to get output.

**Can we run code without clusters?**

In Jupiter notebook we haven’t used this cluster concepts. Instead, as in all normal cases the code will execute in kernel.

**Schema:**

A database schema is a structure that represents the logical storage and structural design of the database.

**Spark RDD:**

Resilient Distributed Dataset

Parallel method

1. **How to initialise spark core in a cluster?** Using SparkContext

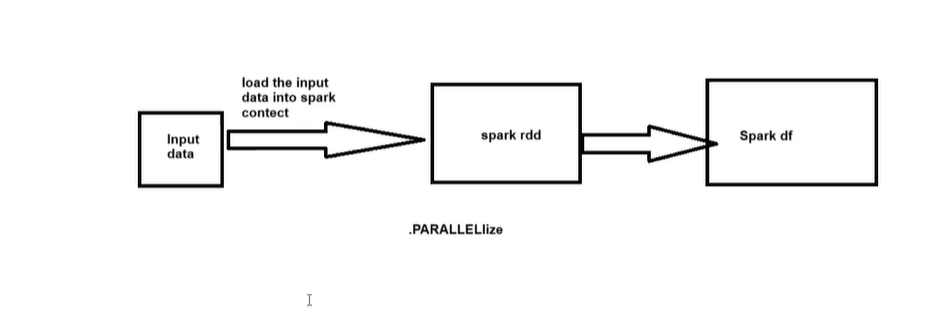
* Importing, Creating., initialising **(this 4 lines is must every code)**

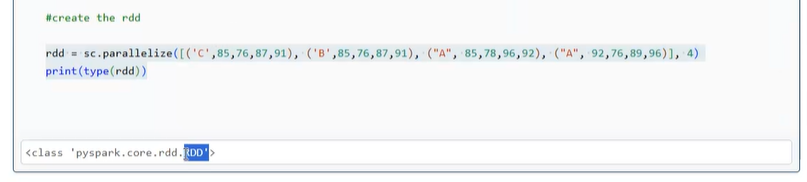
1. **Creating RDD:**

To convert data to data frames(df) by parallelize()

In padas we will just import and read data where it will automatically forms df.

Here 1 step extra i.e we create RDD



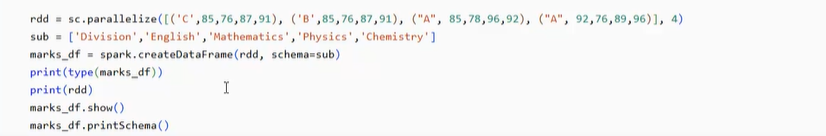


rdd = sc.parallelize([('C',85,76,87,91), ('B',85,76,87,91), ("A", 85,78,96,92), ("A", 92,76,89,96)])

print(type(rdd))

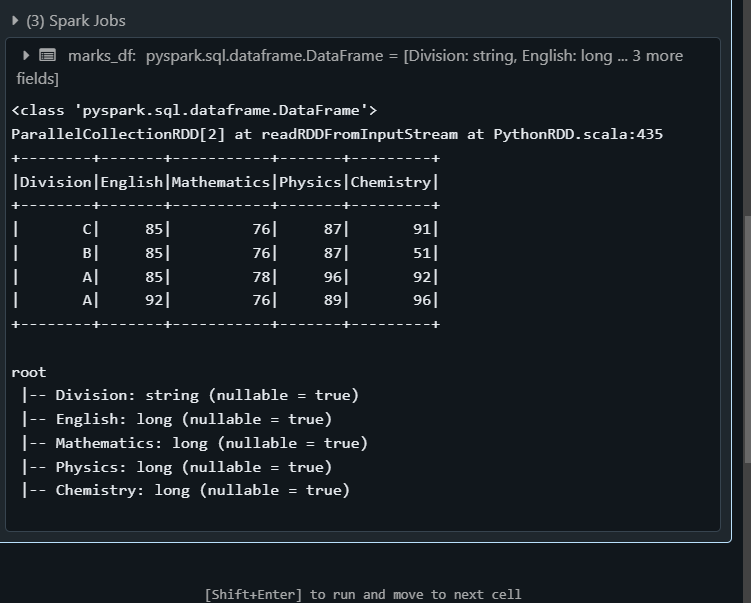
Where type(rdd) is to check rdd creation.

1. **Create Data frame using RRD:**



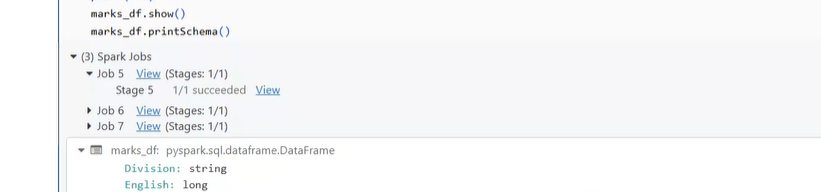
Here schema = sub where sub is the name of our schema.

You can give Schema = Yazh’s data

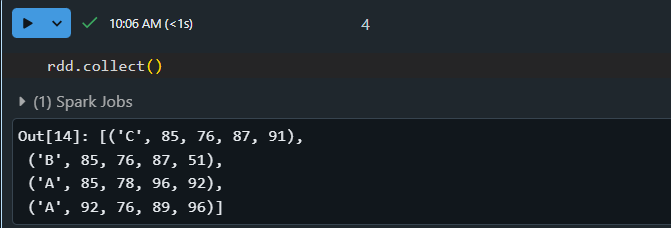
Output:

**Note:** Spark jobs and stages:

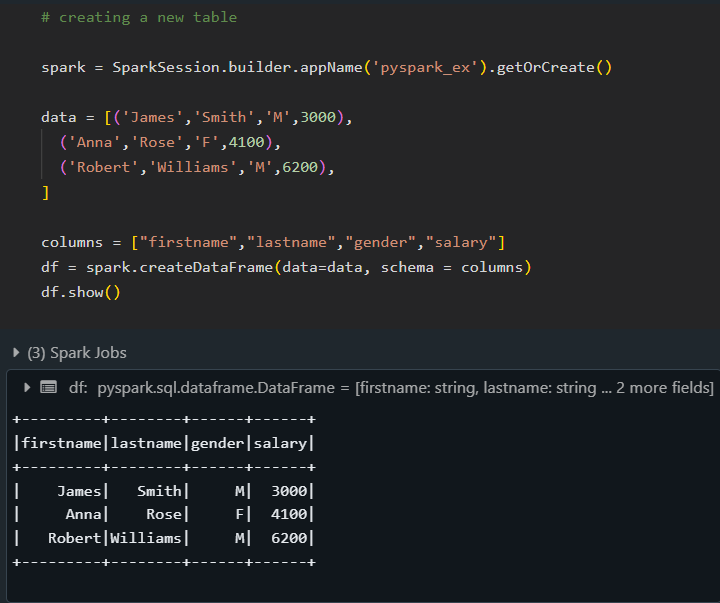
In output you can see a spark Jobs which is auto created where we will get the stages info etc..



* 1. rdd.collect():



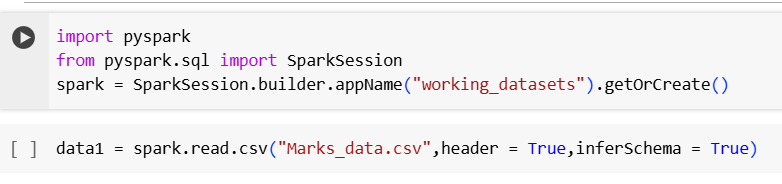
1. **creating a new table**



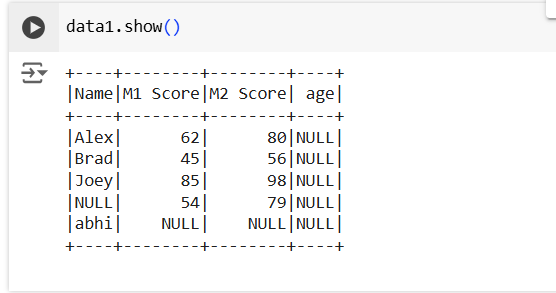
1. **Basic Functions of Spark**

* Selecting, Renaming, Filtering Data in a Pandas DataFrame
* Manipulating, Droping, Sorting, Aggregations, Joining, GroupeBy DataFrames
* Refer – Functions of Spark.pdf (https://drive.google.com/file/d/1KcRVJLROkWEj5KnDgdMTh\_sRePFb--H\_/view?usp=drive\_link)

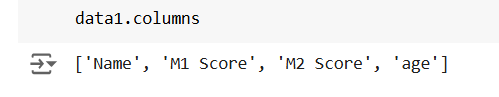
5.1. Importing csv file:



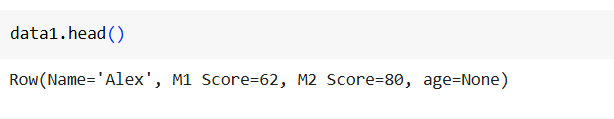
5.2. Show the table:



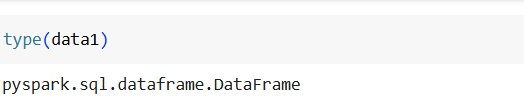
5.3. show the columns list



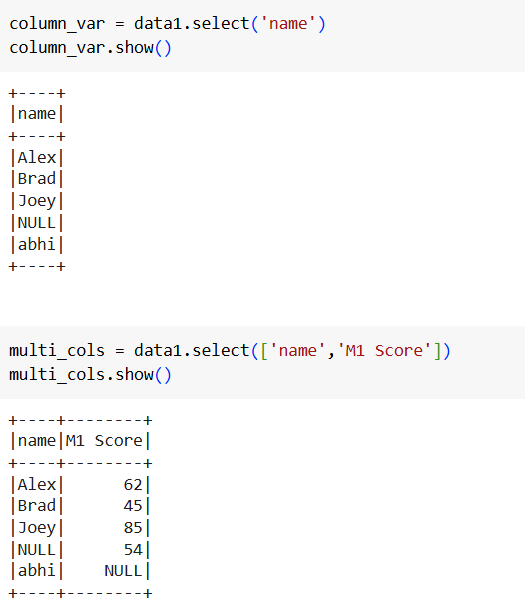
5.4. show head value:



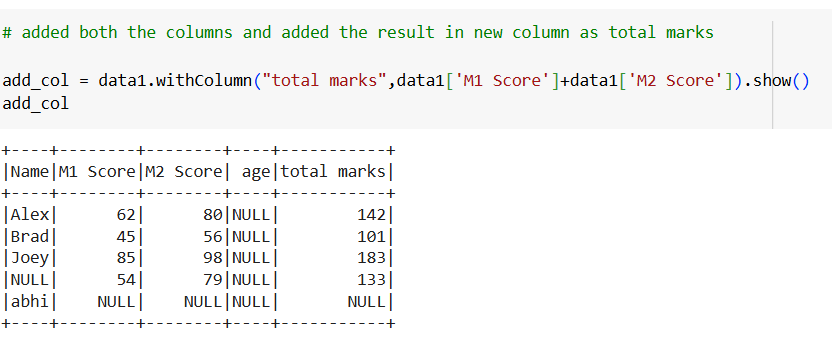
5.5 To display type:



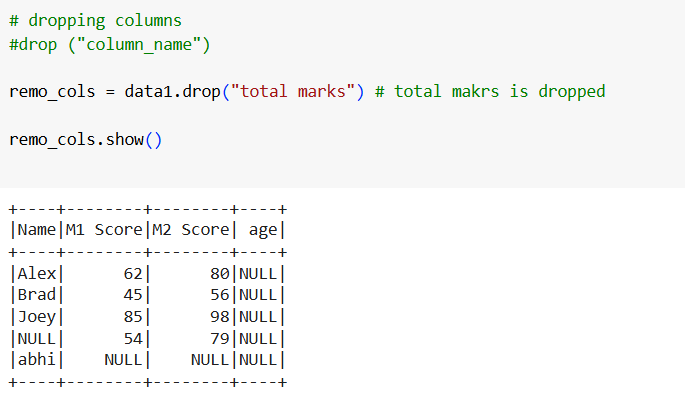
5.6. SELECT:



5.7. ALTER TABLE – ADD column:



5.8. DROP:

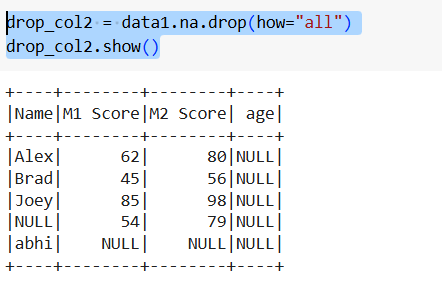


drop\_col2 = data1.na.drop(how="all")

drop\_col2.show()

data1.na.drop(how="all") is used to drop rows from a DataFrame where all the values are null (NaN).

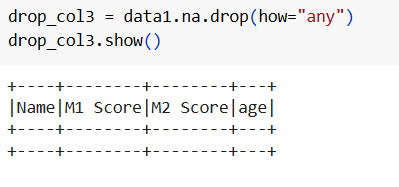
Here we donnot have such column so it doesnot drop anything.



drop\_col3 = data1.na.drop(how="any")

drop\_col3.show()

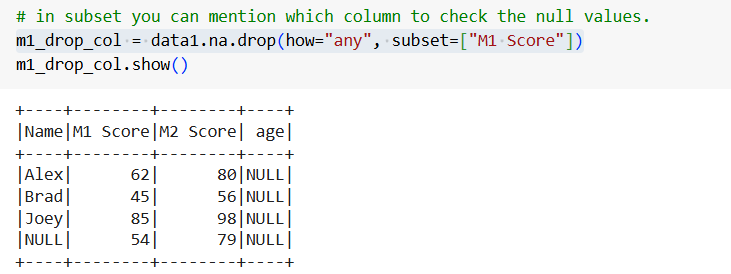
data1.na.drop(how="any") is used to drop rows from a DataFrame even when any one of the values is null (NaN).



m1\_drop\_col = data1.na.drop(how="any", subset=["M1 Score"])

m1\_drop\_col.show()

The code snippet you provided filters out rows from a PySpark DataFrame (data1) that contain null (NaN) values specifically in the column "M1 Score".



5.9. Imputer (for Filling):

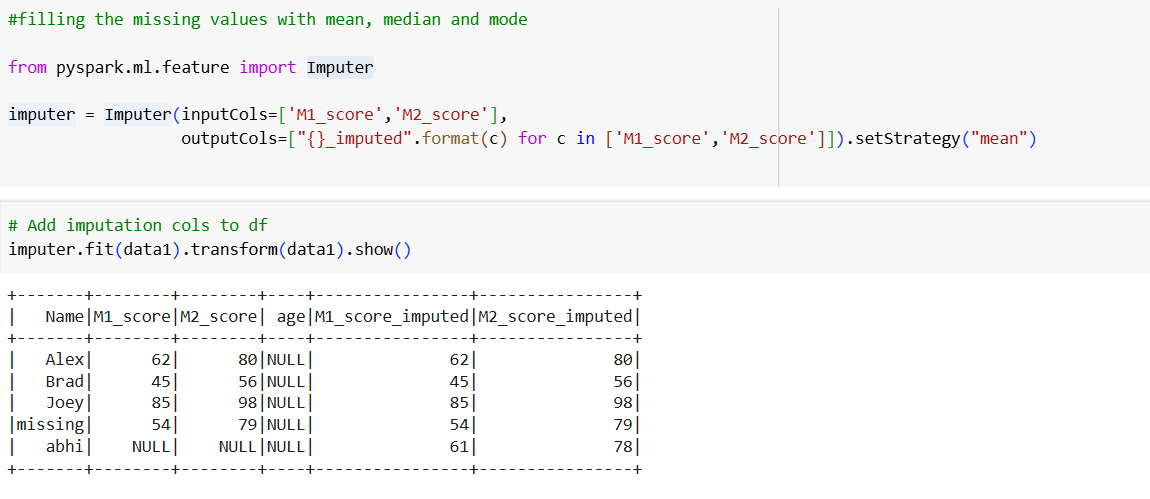
from pyspark.ml.feature import Imputer

imputer = Imputer(inputCols=['M1\_score','M2\_score'],

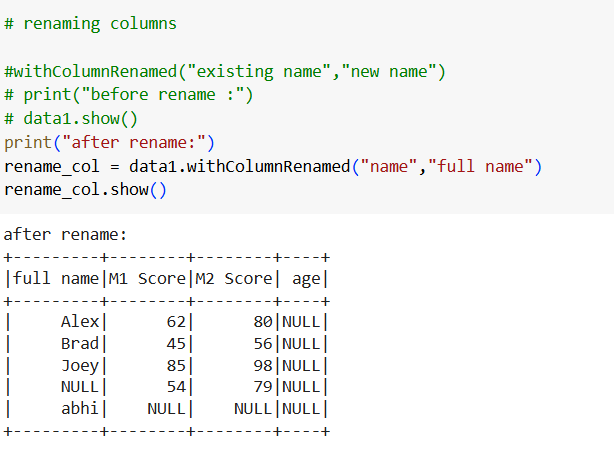
                  outputCols=["{}\_imputed".format(c) for c in ['M1\_score','M2\_score']]).setStrategy("mean")

# Add imputation cols to df

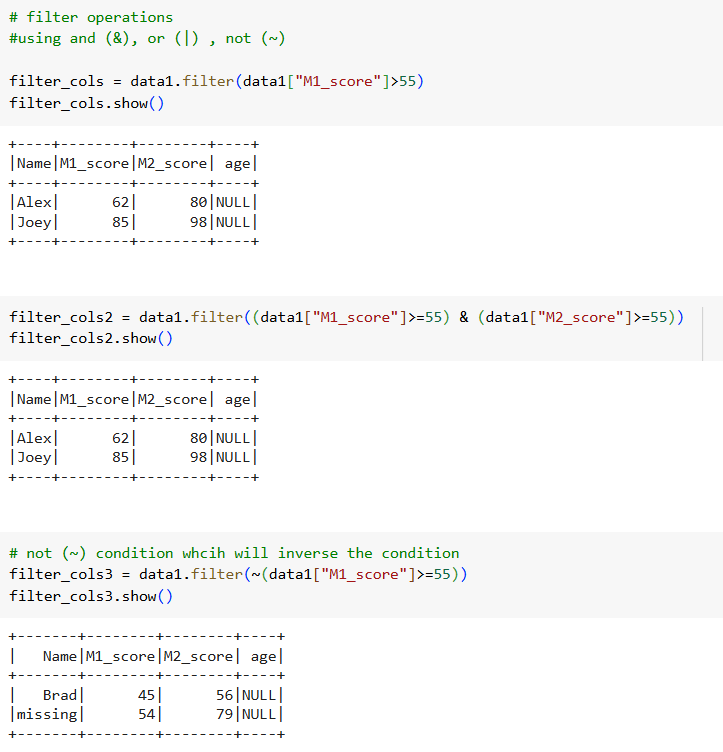
imputer.fit(data1).transform(data1).show()

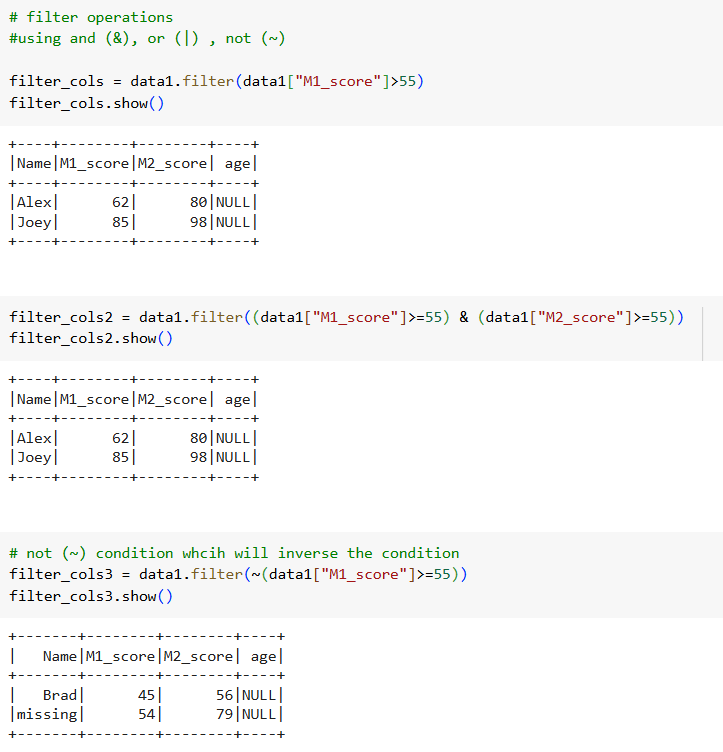


5.10. RENAME:



5.11. FILTER:





5.12. Groupby:

