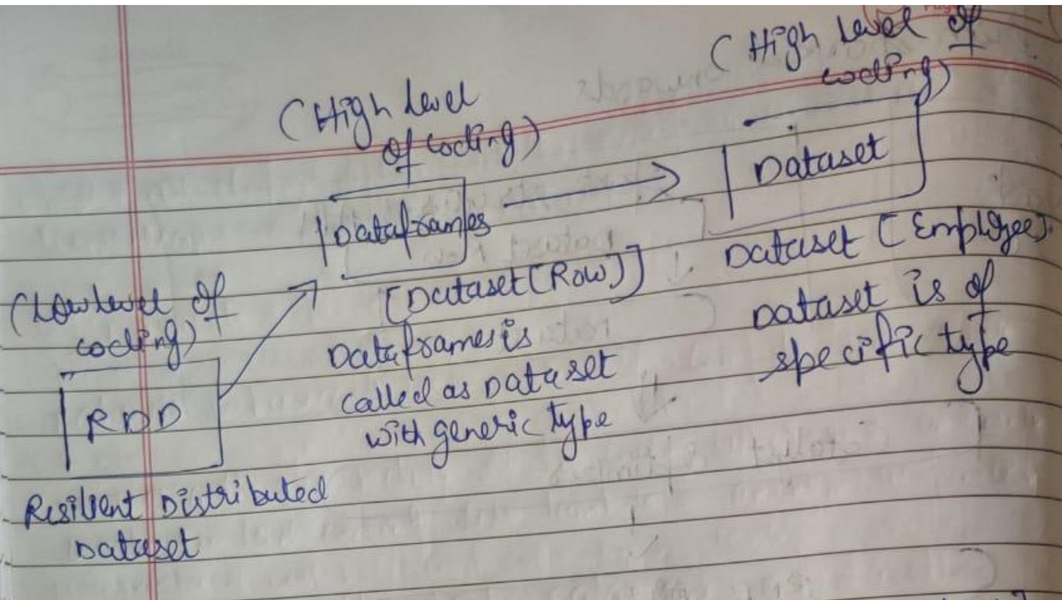

⇒ Advantage of RDD(Resilient Distributed Dataset)

① Type errors are caught at run time (compile type safety)

② Developers have flexibility in writing code, for example anonymous function.

⇒ Disadvantage of RDD, which gave birth to Dataframes.

① Low level of coding is used like using of map, flat map, filter etc. function.

② Optimisation was not there

③ Data was not stored in Tabular format

Developers were facing issue while writing a code for mugging up the low level code

Dataframes

Data was stored in tabular format in the form of table rows & columns like a RDBMS table.

=> Advantage of Dataframes over RDD

1) Code was written in high level construct i.e using where, groupby, select etc function

2) in case of dataframes, since code is written in high level construct first it gets converted to low level coding at driver side, compiler performs some optimisation on it and then it sends to executor for execution. But if we have low level code, it directly sends to executor.

3) Dataframes provide optimisation

=> Disadvantage of Dataframes

1) compile time safety is missing, because of this runtime surprises comes

2) no flexibility provided to developer in terms of like writing anonymous function

To cover the disadvantage of the Dataframes, spark developer thought to convert Dataframes (Df) to Rdd (Rdd) using df.rdd but conversion was not seamless and loosing the optimisation which was given by catalyst optimiser.

so Developer optimised the Dataframes instead of rdd and converted to Dataset

=> Datasets

Datasets is basically take care of compile time safety as well as the flexibility of the code, where we can easily write low level coding like lambda or anonymous function

so if we talk about Dataframes & Dataset there is slight diff. b/w both of them

Dataframes => Dataset [Row]

where Row is nothing but a generic type in which type will be bound at runtime.

Dataset [Employee] -> Dataset

Dataset where type is bound at compile time