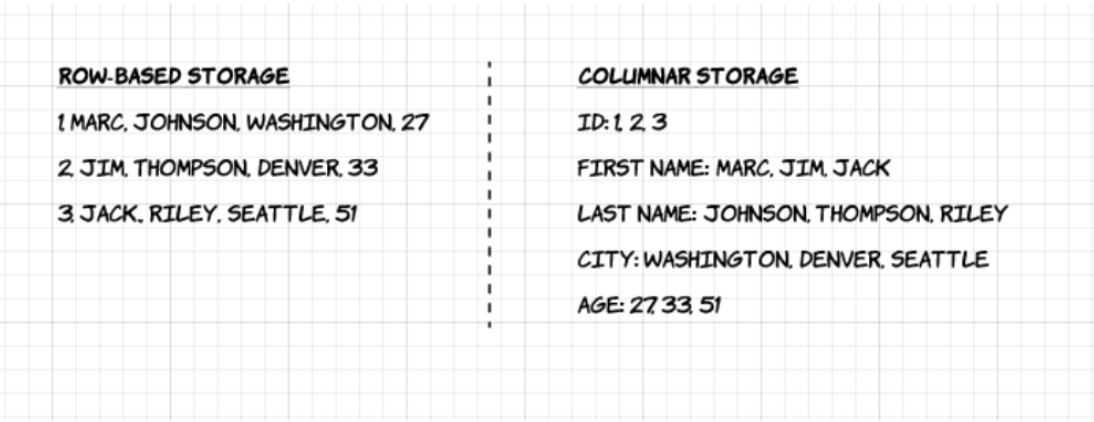
**Apache Parquet Format**

🡪It is most efficient than csv JSON file format.

🡪Parquet file format main motive is to provide snappy compression (fastest data compression developed by google).

🡪Unlike csv, JSON which is row wise file format, parquet is columnar based format.



🡪**A Parquet file contains metadata including schema and structure. Each file stores both the data and the standards used for accessing each record – making it easier to decouple services that write, store, and read Parquet files.**

**🡪Pyspark by default file format is Parquet in all its API’s i.e. python, Scala etc.**

**Advantage of parquet over csv,JSON**

**🡪Performance**

**🡪Compression**

**🡪Schema Evaluation**

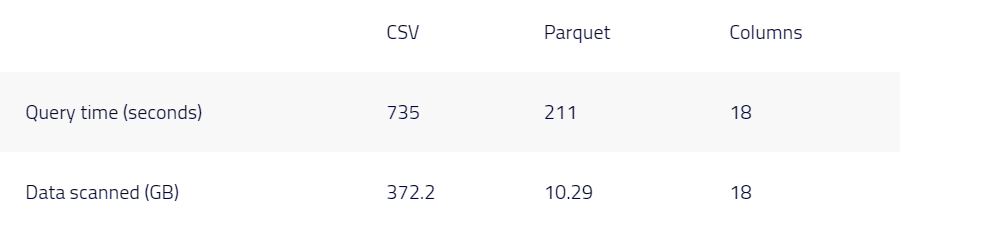
**🡪**Apache Parquet is part of the open-source Apache Hadoop ecosystem.

**When to use Parquet**

🡪When you’re working with very large amounts of data.

🡪When your full dataset has many columns, but you only need to access a subset: Due to the growing complexity of the business data you are recording, you might find that instead of collecting 20 fields for each data event you’re now capturing 100+. While this data is easy to store in a data lake, querying it will require scanning a significant amount of data if stored in row-based formats. Parquet’s columnar and self-describing nature allows you to only pull the required columns needed to answer a specific query, reducing the amount of data processed.

**🡪**When you want multiple services to consume the same data from object storage.



**The above result gives us the clarity of query performance and data processed time of csv and parquet with same number of columns**

**Apache parquet is very optimal solution for data lakes in terms of data processing.**

**Apache Parquet in Pyspark**

**From pyspark.sql import SparkSession**

**Spark=SparkSession.builder.getOrCreate()**

**Parquet\_data=Spark.read.format(“Parquet”).option(headers=True).load(file\_location)**

**Parquet\_data.show()**

**Two options available in parquet are read and Write.**

**csv\_file.write.format(“Parquet”).mode(“OverWrite”).save(location)**