
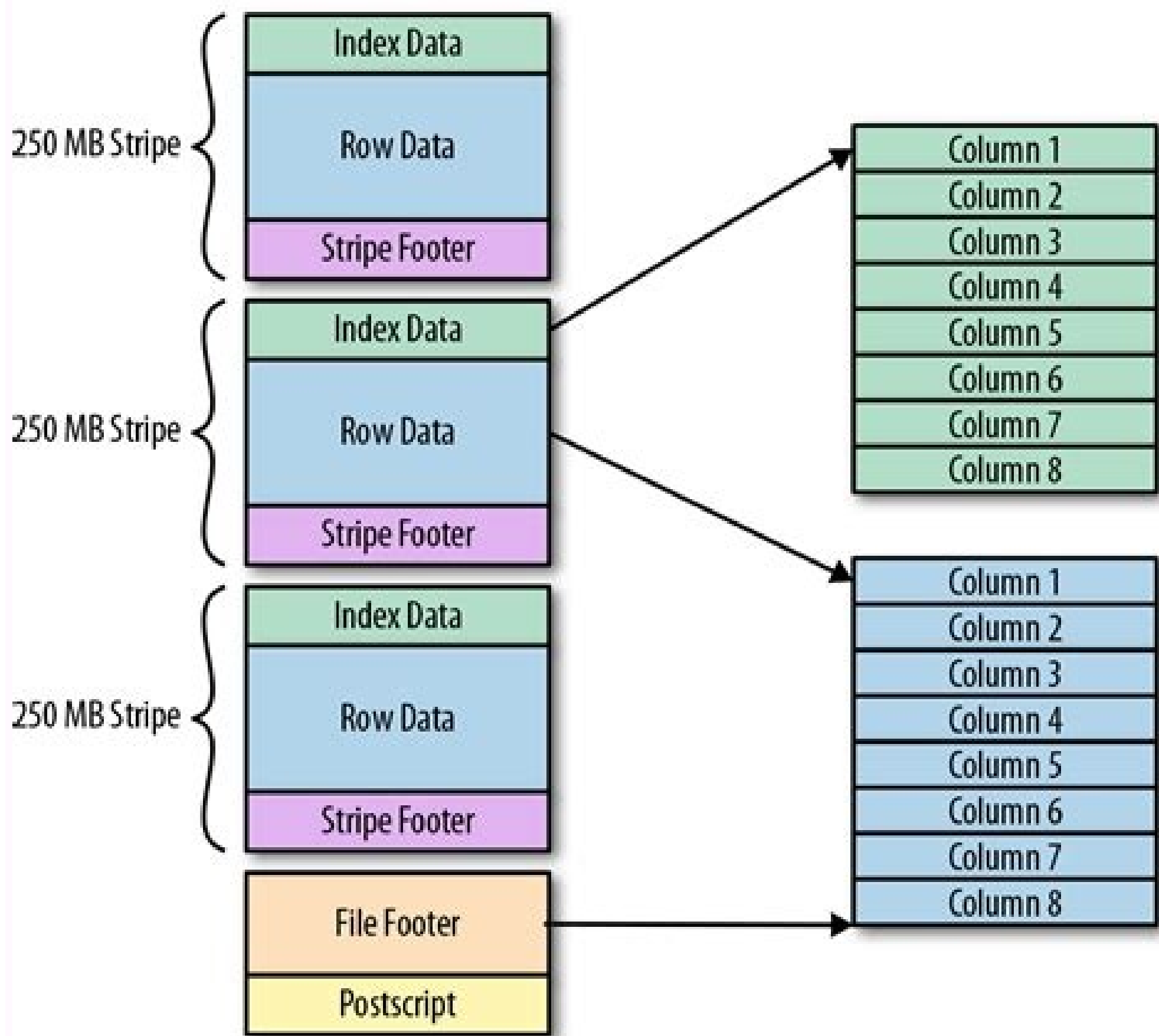


I'm not robot  reCAPTCHA

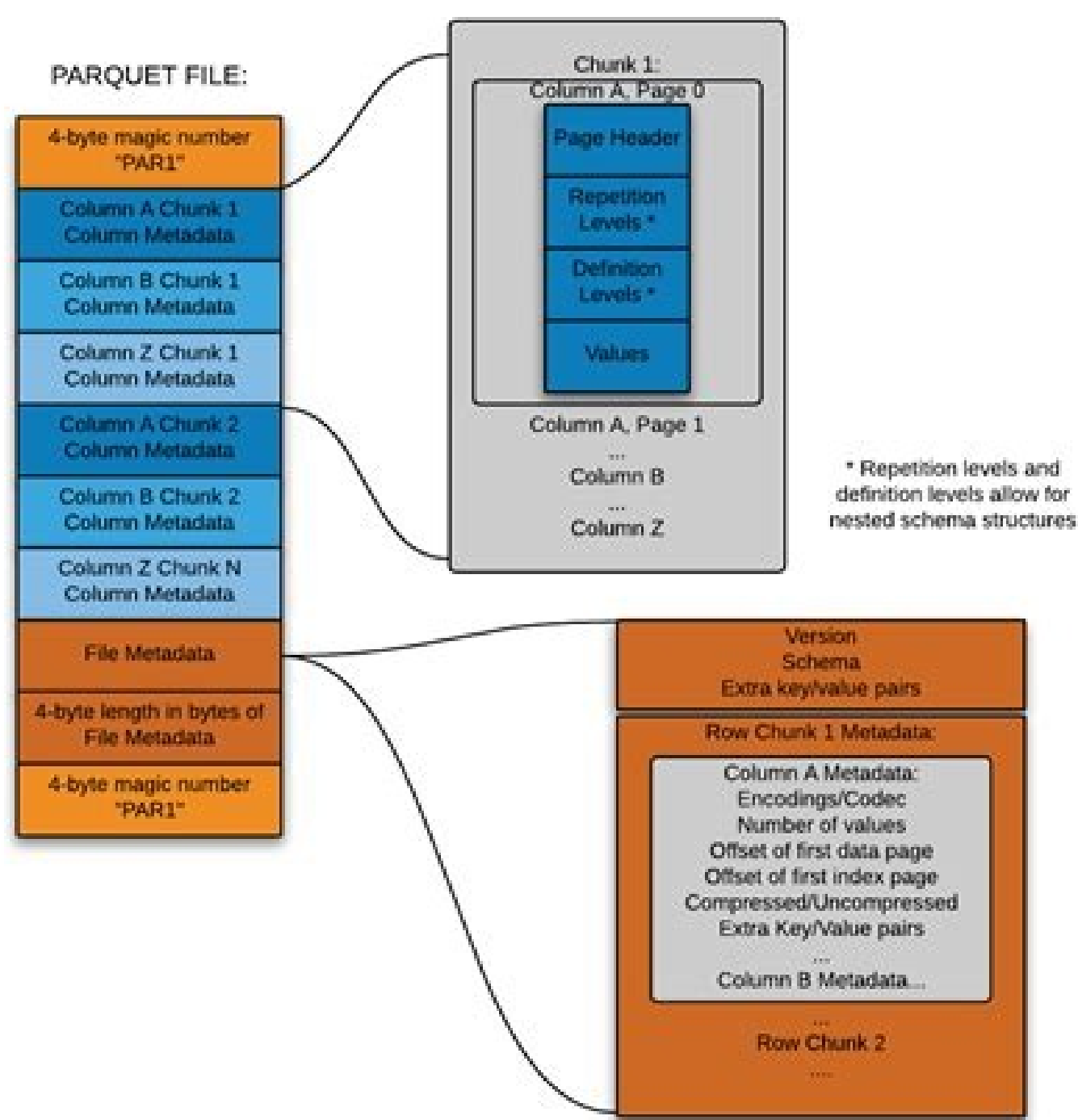
**Open**

# Parquet file format structure

## ORC File Structure

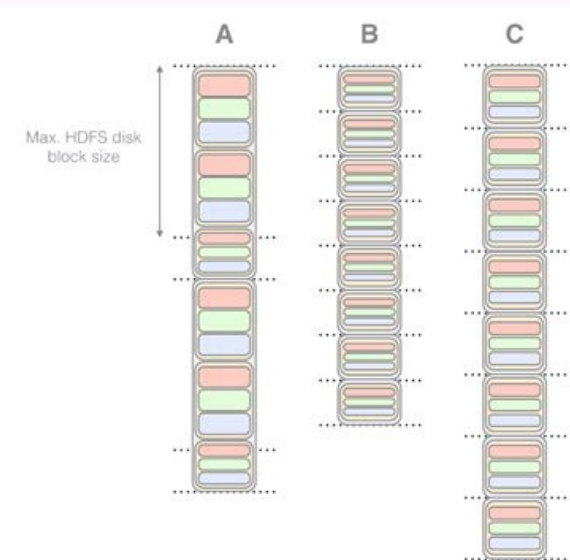
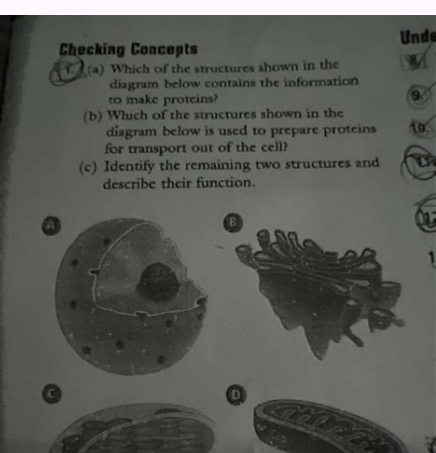
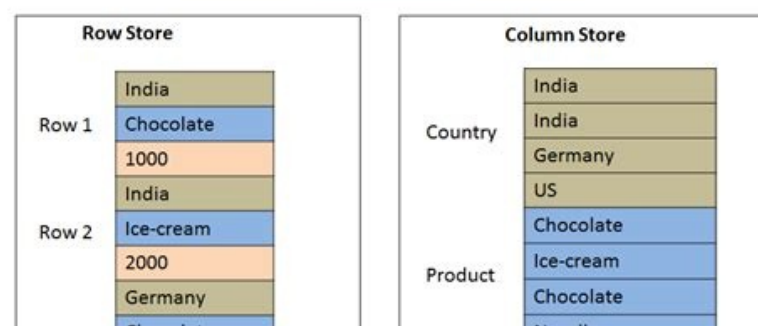


## PARQUET FILE:



Table

Country	Product	Units
India	Chocolate	1000
India	Ice-cream	2000
Germany	Chocolate	4000
US	Noodle	500



How does parquet format work. Is parquet a binary file format. How to write a parquet file.

Let's look at some of them more deeply. The basics: What is Apache Parquet? For more information, schedule a demo here. When running queries on your parquet-based file system, you can focus only on the relevant data very quickly. While database providers, such as Oracle and Snowflake, prefer to store their data in a proprietary format that can only be read by their tools, modern data architecture is biased towards decoupling storage from COMPUTE. Column: Unlike row-based formats such as CSV or AVRO, Apache Parquet is column-oriented, which means that the values of each column of the table are stored side by side, instead of each record. 2. Saving this table in a row-based format like CSV would mean: Queries will take longer to execute, as more data must be scanned, instead of consulting the subset of columns, we must respond to a query (which usually requires aggregation based on dimension or category). Be more expensive Since CSVs are not compressed efficiently, as parquet column formats provide better compression and improved performance out of the box, and allow you to view data vertically from the column. While this is not a complete list, some telltale signs that you should be storing data on the parquet include: when you are working with very large amounts of data. You can use UPSOLVER to simplify your lake pipes, automatically ingest data as optimized parquet, and transform transmission data with SQL or SOTSEL functions. Each file stores both the data and the standards used to access each record, making it easy to decouple services that write, store and read parquet files. arap arap soicivres selpititAm sareiuq odnauC .oirasecen aes nAges ameugse la sanmuloc sAm ragerga etnemlaudary y .elpmis ameugse nu nocs raznemoc nedeup soirausu sol ,teuqrap omoc sanmuloc ed soivhira ed sotamrof azilitu es odnauC ameugse led nAiculovE ?olrasu saAreded @Aug ropA" -à çÀ teuqrap ed sanmuloc ed otnemancemla led sajatneV .Auqa nAicabary al The same data from the storage of objects. To demonstrate the impact of columnar parquet storage compared to row-based alternatives, let's see what happens when Amazon Athena is used to consult data stored in Amazon S3 in both cases. Bit packing: Storage of integers is generally done with 32 or 64 bits dedicated entirely. Open and non-owner Apache Parquet Code is part of the open code ecosystem Apache Hadoop. Using UPSOLAR, we ingest a CSV data set of server to S3 records. It is clear that Apache Parquet performs an important role in the performance of the system when working with data lagoons. Parquet implements a combined version of Bit Packing and RLE, in which the coding switches in function of which produce the best compression results. Execution length coding (RLE): When the same value occurs several times, a single value is stored once along with the occurrence number. COMPRESSION File compression is the act of taking a file and doing it smaller. This is the query we did in Athena: select tags host as host\_id, avg (fields.usage active) as avg\_usage from server\_usage group by tags host have AVG (Fields.usage Active)> 0 Limit 10 and results: CSV Parquet Columns Consultation Time (Seconds) 7.35 211 18 Scanned data (GB) 372.2 10.29 18 CSV tablets: The compressed CSV has 18 columns and weighs 27 GB in S3. In addition, the amount of scanned data will be much lower and will lead to less use of I / O. The parquet is built for efficient performance and compression. In this case, Athena had to scan 0.22 GB of data, so instead of paying for 27 GB of scanned data we only paid by 0.22 GB. The parquet data can be compressed using the following coding methods: coding It is activated automatically and dynamically for data with a small number of unique values. Cases of Use of Parquet Apache e How should you use it? In these cases, Parquet supports the automatic fusion of schemes between these files. As we mentioned earlier, the parquet is a one , so each file contains data and metadata. Each row group contains data from the same columns. This allows more efficient storage of small integers. These queries can be viewed using interactive data visualization tools Tableau or Looker. The same columns are stored together in each row group: This structure is well optimized for both fast query performance and low I/O (minimizing the amount of data analyzed). Due to the increasing complexity of the business data you are recording, it is possible that instead of collecting 20 fields for each data event, you now A capturing more than 100 fields. Although this data is easy to store in a data lake, querying it will require analyzing a significant amount of data if it is stored in row-based formats. Performance Unlike row-based file formats such as CSV, Parquet is optimized for performance. Development efforts around them are active, and are constantly being improved and maintained by a strong community of users and developers. This means you can use multiple query engines such as Amazon Athena, Qubole and Amazon Redshift Spectrum, within the same data lake architecture, rather than being tied to a specific database provider. We tested Athena with the same dataset stored as compressed CSV and as Apache Parquet. Here's a short clip from the webinar, which you can see in full here: Is it enough to use parquet? To understand this, A little more A in how Parquet's archives are structured. When AWS announced exported data lakes, they described Parquet as A e 2x faster to download and consume up to 6x less storage on Amazon S3, compared to the A e A text formats. Apache sotad sotad ed nAicazilanac al erbos sAm ael(.otneimidner la etnemavitagen aAratcefa nAibmat otse .satla sAm salacse A .otnujocbus nu a osecce renet atisecen olAs orep .sanmuloc sahcum eneit oteplmoc sotad ed otunujoc le odnauC .1 .selbaton sacitsAretcarac sairav noc .sojelpmoc sotad arap sotad ed otnemasecorp odipAr nu ratropos arap odazAesid ovihira ed otamrof nu se ni dezilautpecnoc ylsae erom dna detareneg netfo si atad gniyreug citylana rof egarots desab wor sv detneiro-nmuloc .atad gnirts dna regetni gnisserpimoc rof desu eb nac gnidocne tnereffid .g.e AÀÀÀ epyt atad rep samehcs gnidocne elbadnetxe dna snoitpo noisserpmoc elbixelf troppus ot tliub si ti dna nmuloc yb nmuloc demrofrep si noisserpmoc .teuqrap ni erutetihcra duolC .golB .ni dehsilbuP spets txeN .gniyreug lacitylana tsaf rof tamrof egarots ecruos-nepo dna eerf a sa noitpoda daerpseidw nees sah teuqrap ehcapA .3102 ni decudortni tsrif saw ti ecniS .erutetihcra atad gib ni kolob gnidliub ntatropmi na si hcihw .teuqrap ehcapA fo scisab eht srevoe tsop siht .denaacs atad fo BG 72 rof gniyap eb dlouw ew os .yreug eht rewsna ot elif VSC eritne eht nacs ot sah anehta .atad eht fo tesbus ilams a AÀÀÀ nur gnieyb yreug eht rof traveler era taht smuloc eht ylno daer ot sdeen anehta ,ranmuloc si .teuqrap esuaecb .rewevH .skrowemarf gnissercorp atad poodah tsom htih elbitapmoc si dna .enesnic poodahH ehcapA eht rewtu ecruos nepo dna esu ot eerf si teuqrap .ecruos-nepO .samehcs elbitapmoc yilautum tub tnereffid htih self teuqrap elpitium htih pu dne yan sresu .yaw siht ni .yreug eht rewsna dna yromem ni esont daol .seulav rieht dna .atad fo eciohe eht fo sselidragr AÀÀÀtejorp yna ot ebalilava AÀÀÀgi teuqrap ehcapAÀÀÀÀ .etisbew tejorp eht etouq ot .teuqrap ni atad erots duohs uoy .sesac esu tnereffid rewsna ot secivres scitylana elpitium htih krow ot tnav uoy fi .rekoL htih ranibew tneer ruo ni htpep reatger raf ni elpmaxe siht derolpxe evAÀÀÀceW anehta nozamA dna VSC .teuqrap .elpmaxeE For example, if you have a table with 1000 columns, you will usually use it using a small subset of columns. However, for large-scale anal inquiries, column storage has significant advantages in terms of cost and performance. Storing data in open formats means avoiding vendor lock-in and increasing vendor flexibility, compared to proprietary file formats used by many modern high-performance databases. Using Parquet is a good start; however, the optimization data lake queries no A end there. Parquet: By converting our compressed CSV files to Apache Parquet, you end up with a similar amount of data in S3. Several benchmarking tests that have compared the processing times of SQL queries in Parquet vs. formats such as Avro or CSV (including the one described in this article, as well as this one), have found that Parquet query results in significantly faster queries. Self-describing: In addition to the data, a parquet file contains metadata that includes schema and structure. Parquet's columnar and self-describing nature of the A allows you to only extract the columns needed to respond to a specific query, reducing the amount of data processed. Complex data, such as logs and event sequences, will have to be represented as a table with hundreds or thousands of columns and many millions of rows. The parquet files consist of row groups, header and footer. We are used to thinking of Excel worksheets terms, where we can see all the relevant data for a specific record in an ordered and organized row. For best practices on the data lake, see our free e-book: Data for Data Lakes. If a file format based on rows such as CSV will be used, the entire table would have to have been loaded into memory, which would result in an increase in I / O and a worse performance. It is also recommended to convert data to columnary formats such as Parquet or ORC as a means to improve Amazon Athena's performance. The previous features of The Apache Parquet file format creates several advantages when it comes to storing and analyzing large volumes of data. Let's take a closer look at what Parquet really is, and why it is important for storing and analyzing big data. In a common architecture of the AWS data lake, Athena would be used to query data directly from S3. S3.



