Some things to consider when choosing the format are:

* The structure of your data: Some formats accept nested data such as JSON, Avro or Parquet and others do not. Even, the ones that do, may not be highly optimized for it. Avro is the most efficient format for nested data, I recommend not to use Parquet nested types because they are very inefficient. Process nested JSON is also very CPU intensive. In general, it is recommended to flat the data when ingesting it.
* Performance: Some formats such as Avro and Parquet perform better than other such JSON. Even between Avro and Parquet for different use cases one will be better than others. For example, since Parquet is a column based format it is great to query your data lake using SQL whereas Avro is better for ETL row level transformation.
* Easy to read: Consider if you need people to read the data or not. JSON or CSV are text formats and are human readable whereas more performant formats such parquet or Avro are binary.
* Compression: Some formats offer higher compression rates than others.
* Schema evolution: Adding or removing fields is far more complicated in a data lake than in a database. Some formats like Avro or Parquet provide some degree of schema evolution which allows you to change the data schema and still query the data. Tools such [Delta Lake](https://delta.io/) format provide even better tools to deal with changes in Schemas.
* Compatibility: JSON or CSV are widely adopted and compatible with almost any tool while more performant options have less integration points.

File Formats

* CSV: Good option for compatibility, spreadsheet processing and human readable data. The data must be flat. It is not efficient and cannot handle nested data. There may be issues with the separator which can lead to data quality issues. Use this format for exploratory analysis, POCs or small data sets.
* JSON: Heavily used in APIs. Nested format. It is widely adopted and human readable but it can be difficult to read if there are lots of nested fields. Great for small data sets, landing data or API integration. If possible convert to more efficient format before processing large amounts of data.
* Avro: Great for storing row data, very efficient. It has a schema and supports evolution. Great integration with Kafka. Supports file splitting. Use it for row level operations or in Kafka. Great to write data, slower to read.
* Protocol Buffers: Great for APIs, especially for [gRPC](https://grpc.io/" \t "_blank). Supports Schema and it is very fast. Use for APIs or machine learning.
* Parquet: Columnar storage. It has schema support. It works very well with Hive and Spark as a way to store columnar data in deep storage that is queried using SQL. Because it stores data in columns, query engines will only read files that have the selected columns and not the entire data set as opposed to Avro. Use it as a reporting layer.
* ORC: Similar to Parquet, it offers better compression. It also provides better schema evolution support as well, but it is less popular.

**Csv, Json, Avro are row based format while parquet, orc are column based.**