Data Engineering

File Types

Cheat Sheet

Cloud with Garvit

1. CSV (Comma-Separated Values)

- **Description**: Simple text files where each line is a data record, and fields are separated by commas.

- **Pros**:
  - Human-readable, easy to parse.
  - Compatible with many tools (Excel, SQL databases, etc.).

- **Cons**:
  - No support for complex data structures.
  - Inefficient for large data (not compressed).

- **Use Cases**: Data exchange, simple storage, and loading into RDBMS.

2. TSV (Tab-Separated Values)

- **Description**: Similar to CSV but uses tabs as delimiters.

- **Pros**: Easier to parse in cases where commas are part of the data.

- **Cons**: Still lacks complex structure support.

- **Use Cases**: When data contains commas, lightweight data storage.

3. JSON (JavaScript Object Notation)

- **Description**: Text-based format that represents structured data as key-value pairs.

- **Pros**:
  - Supports nested data structures (arrays, objects).
  - Widely supported and readable.

- **Cons**:
  - Can be verbose and less efficient for large datasets.
  - Not schema-enforced, which can lead to data inconsistency.

- **Use Cases**: Web APIs, NoSQL databases, log files, semi-structured data.

4. XML (Extensible Markup Language)

- **Description**: Text-based format that uses tags to represent structured data.

- **Pros**:
  - Allows custom schema and validation (XSD).
  - Supports complex and nested data.

- **Cons**:
  - Verbose, leading to large file sizes.
  - Parsing is resource-intensive.

- **Use Cases**: Data interchange between systems, legacy applications.

5. Parquet

- **Description**: Columnar storage format optimized for read-heavy workloads.

- **Pros**:
  - Columnar storage enables efficient data retrieval.
  - Supports compression, making it storage-efficient.
  - Schema support provides data consistency.

- **Cons**: Not human-readable.

- **Use Cases**: Big data processing in Spark, Hadoop, and Azure Data Lake, and analytics workloads.

6. Avro

- **Description**: A row-based binary storage format optimized for write-heavy workloads.

- **Pros**:
  - Fast serialization/deserialization.
  - Embedded schema, which facilitates data versioning.

- **Cons**: Not human-readable, less efficient for columnar storage.

- **Use Cases**: Event streaming (Kafka), NoSQL data storage, schema evolution handling.

7. ORC (Optimized Row Columnar)

- **Description**: Columnar storage format designed for large datasets, primarily in Hadoop.

- **Pros**:
  - High compression rates.
  - Fast read/write capabilities for Hive and big data tools.

- **Cons**: Limited support outside of Hadoop ecosystems.

- **Use Cases**: Hive, big data analytics, Hadoop environments.

8. Excel (XLS/XLSX)

- **Description**: Proprietary spreadsheet formats with support for tables, formulas, and charts.

- **Pros**:
  - Easy to use for data entry and simple analysis.
  - Can handle basic visualization.

- **Cons**:
  - Not suitable for large datasets.
  - Limited support in big data tools.

- **Use Cases**: Data entry, small datasets, and quick analysis.

9. HDF5 (Hierarchical Data Format)

- **Description**: Binary format that stores data in a hierarchical structure, suitable for large scientific datasets.

- **Pros**:
  - High performance for large, multidimensional data.
  - Supports complex data types.

- **Cons**: Requires specific libraries for reading/writing.

- **Use Cases**: Scientific computing, machine learning, and neural network training data.

10. TXT (Plain Text)

- **Description**: Unstructured format, often used for logs or simple data storage.

- **Pros**:
  - Human-readable and easily modified.
  - Simple and portable.

- **Cons**:
  - No structure or schema.
  - Not storage-efficient.

- **Use Cases**: Logs, simple data storage, unstructured data.

11. SQL (Structured Query Language) Files

- **Description**: Contains SQL commands for defining or querying relational databases.

- **Pros**: Allows direct use of SQL for data manipulation.

- **Cons**: Only useful for SQL-compatible systems.

- **Use Cases**: Database backup, migration scripts, data extraction from RDBMS.

12. Binary Format

- **Description**: Low-level format, optimized for performance but not human-readable.

- **Pros**: Fast read/write speeds and efficient storage.

- **Cons**: Not portable or readable.

- **Use Cases**: System-specific data storage, embedded systems, certain big data applications.

13. Image Formats (JPEG, PNG, TIFF)

- **Description**: Used for storing visual data.

- **Pros**: Common in industries needing image processing.

- **Cons**: Not structured for relational data or analytics.

- **Use Cases**: Medical imaging, deep learning (image recognition).

14. Audio/Video Formats (MP3, WAV, MP4)

- **Description**: Stores audio and video data.

- **Pros**: Useful for multimedia and ML applications.

- **Cons**: Requires specialized processing tools.

- **Use Cases**: Audio analysis, video streaming, speech recognition.

15. Protocol Buffers (Protobuf)

- **Description**: Language-neutral format by Google, optimized for serialization and deserialization.

- **Pros**:
  - Highly efficient.
  - Supports schema evolution.

- **Cons**: Binary format, requires Protobuf libraries.

- **Use Cases**: High-performance data exchange, mobile applications, streaming data.

16. YAML (Yet Another Markup Language)

- **Description**: Human-readable format often used for configuration files.

- **Pros**:
  - Easy to read and write.
  - Supports complex data structures.

- **Cons**: Limited support for large datasets or big data.

- **Use Cases**: Configuration files, data exchange for small data applications.

Summary Table

Format	Structure	Pros	Cons	Common Uses
CSV/TSV	Flat	Simple, portable	No support for nested data	Data exchange, simple storage
JSON	Hierarchical	Flexible, semi-structured	Inefficient for large datasets	APIs, NoSQL
XML	Hierarchical	Custom schema support	Verbose	Data interchange, legacy
Parquet	Columnar	High read performance	Not human-readable	Big data, analytics
Avro	Row-based	Fast, schema support	Limited to row-based use	Streaming, schema evolution
ORC	Columnar	Compressed, optimized for Hadoop	Limited to Hadoop	Hadoop, big data analytics
Excel	Flat	User-friendly	Limited scalability	Data entry, small datasets
HDF5	Hierarchical	High performance for large data	Specialized libraries needed	Scientific, ML training data
TXT	None	Simple, human-readable	No structure	Logs, unstructured data
SQL	Structured	SQL-compatible	RDBMS-dependent	Data migration, DB scripts
Binary	None	Efficient storage	Not human-readable	Embedded systems, big data
Image	None	Industry-standard formats	Not structured	Medical, image processing
Audio/Video	None	Audio and video compatibility	Specialized tools required	ML, audio, video analytics
Protobuf	Binary	High performance, schema support	Requires libraries	Mobile apps, streaming
YAML	Hierarchical	Readable, flexible	Limited for big data	Config files, small data apps