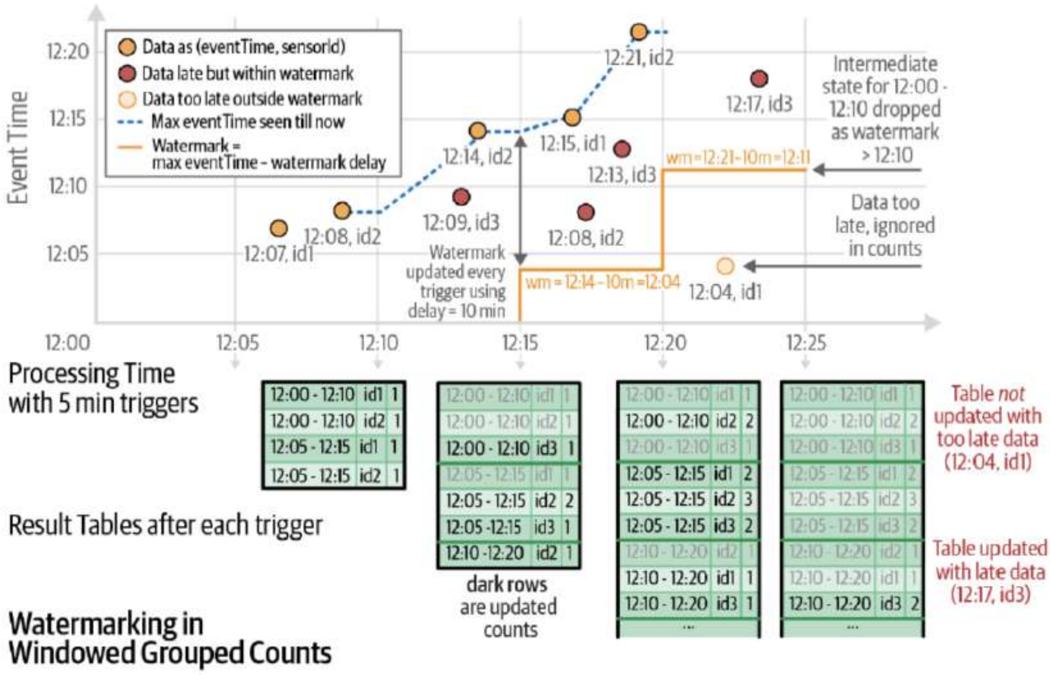
Handling Late Data with Watermarks

```
(sensorReadings
  .withWatermark("eventTime", "10 minutes")
  .groupBy("sensorId", window("eventTime", "10 minutes", "5 minutes"))
  .count())
```



Data lakes

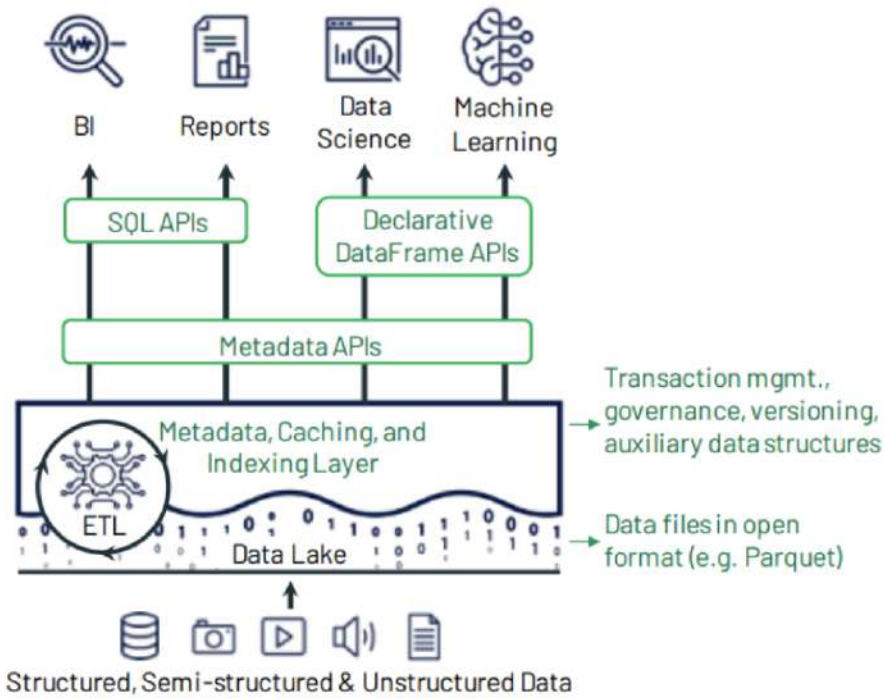
- decouples the distributed storage system from the distributed compute system
  - Allows each system to scale out as needed by the workloads
- Organizations build their data lakes by independently choosing
  - Storage system: HDFS, S3, Cloud and etc.
  - File format:
    - Structured: Parquet, ORC
    - semi-structured: JSON
    - unstructured formats: text, images, audio, video
  - Computing / Processing engine(s):
    - batch processing engine: Spark, Presto, Apache Hive
    - stream processing engine: Spark, Apache Flink
    - machine learning library: Spark MLlib, scikit-learn, R

Data Lakes

- Pros
  - Flexibility on choosing storage, data format and processing engines
  - A much cheaper solution than databases → explosive growth of the big data ecosystem

- Cons:
  - Fail to provide ACID guarantees
  - Building and maintaining an effective data lake requires expert skills
  - Easy to ingest data but very expensive to transform data to deliver business values
  - Data quality issues due to the lack of schema enforcement

Data Lakehouse implementation

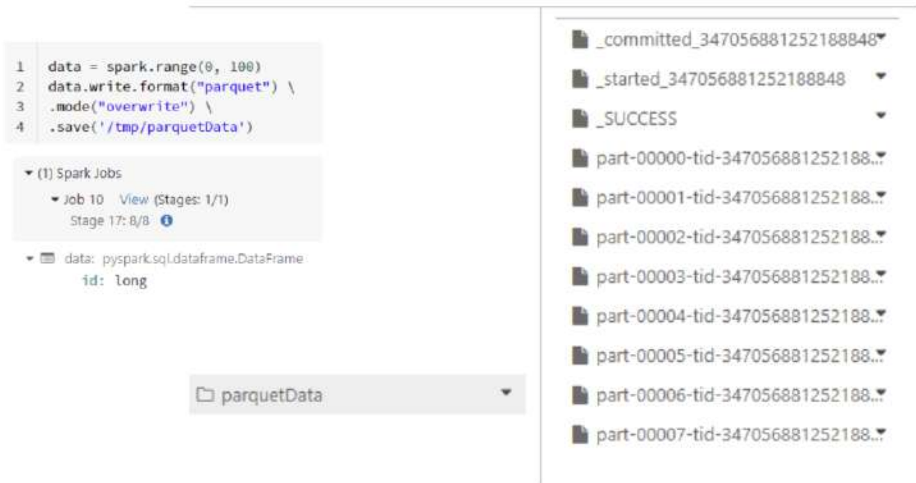


Delta Lake

- the metadata, caching and indexing layer on top of a data lake storage that provides an abstraction level to serve ACID transactions and other management features
  - Transactional ACID guarantees
  - Full DML (Data Manipulation Language) support
  - Audit History
  - Unification of batch and streaming into one processing model
  - Schema enforcement and evolution
  - Rich metadata support and scaling

Delta Lake Format

- a standard Parquet file with additional metadata
- Parquet Files
  - Column oriented: perform compression on a column-by-column basis
  - Open source
  - Self-describing: actual data + metadata (schema & file structure)



The Delta Lake Transaction Log (DeltaLog)

- The transaction log is an ordered record of every transaction made against a Delta table since it was created.
- It acts as a single source of truth and tracks all changes made to the table.
- The main goal is to enable multiple readers and writers to operate on a given version of a dataset simultaneously.
- It is at the core of many important features
  - ACID transactions
    - Spark looks at the transaction log to get the latest version of the table
    - If an operation is not recorded in the transaction log, it never happened.

  - Scalable metadata handling
  - Time travel

Five Steps to Define a Streaming Query

- Step 1: Define input sources
- Step 2: Transform data
- Step 3: Define output sink and output mode
  - Output writing details (where and how to write the output)
  - Processing details (how to process data and how to recover from failures)
- Step 4: Specify processing details
  - Triggering details: when to trigger the discovery and processing of newly available streaming data.
  - Checkpoint Location: store the streaming query process info for failure recovery
- Step 5: Start the query