# Explanation of test\_cases.py and run\_test.py

## test\_cases.py

The test\_cases.py file is designed to test the ETL pipeline for a given configuration and Spark environment.  
 It is structured using the Python unittest framework, and the primary purpose is to validate each stage  
 (Extract, Transform, and Load) of the ETL process.  
  
 Key Components:  
 - The class `ETLTestCases` inherits from `unittest.TestCase`, enabling structured testing.  
 - Class variables `spark`, `config`, `file\_paths`, and `logger` are used to inject dependencies like  
 Spark session, configuration, data file paths, and logging functionality.

### load\_sample\_data()

Purpose:  
 - Loads sample data from CSV files into Spark temporary views for testing.  
  
 Steps:  
 1. Validates that file paths are initialized and not empty.  
 2. Iterates through the provided file paths to load CSV files using Spark's DataFrame API.  
 3. Creates temporary views for the loaded data to simulate the actual database environment.  
  
 Exception Handling:  
 - Raises a `FileNotFoundError` if any file path is missing.  
 - Raises a `RuntimeError` if loading fails for a specific table.

### validate\_sql\_syntax(query)

Purpose:  
 - Validates the SQL syntax of a query using Spark's `EXPLAIN` command.  
  
 Steps:  
 1. Runs the `EXPLAIN` command for the given query to check its validity.  
 2. Returns True if the query is valid; otherwise, captures and returns the error message.  
  
 Benefits:  
 - Catches SQL syntax errors early, ensuring only valid queries are executed.

### test\_extract\_data\_lumi()

Purpose:  
 - Simulates the "Extract" phase of the ETL process by executing extract queries on temporary views.  
  
 Steps:  
 1. Retrieves extract queries from the provided configuration.  
 2. Validates the SQL syntax for each extract query.  
 3. Executes each query and creates a temporary view for the extracted data.  
  
 Exception Handling:  
 - Raises a `RuntimeError` if SQL syntax validation fails or if query execution fails.

### test\_etl\_pipeline()

Purpose:  
 - Validates the entire ETL pipeline, including Extract and Transform stages.  
  
 Steps:  
 1. Loads sample data into temporary views.  
 2. Executes extract queries and validates that the expected views are created.  
 3. Executes transform queries and validates that the expected views are created.  
  
 Exception Handling:  
 - Raises an assertion error if any expected view is not created.  
 - Validates SQL syntax before execution to catch errors early.

### test\_load\_simulation()

Purpose:  
 - Simulates the "Load" phase by validating that the final transformed data is ready for loading.  
  
 Steps:  
 1. Retrieves the load configuration (e.g., target table name and DataFrame name).  
 2. Queries the final DataFrame and ensures it contains data.  
 3. Logs the number of rows in the final DataFrame for verification.  
  
 Exception Handling:  
 - Raises an assertion error if the final DataFrame is empty or inaccessible.

## run\_test.py

The run\_test.py file serves as the entry point for running the test cases defined in test\_cases.py.  
 It sets up the testing environment and triggers the tests.  
  
 Key Components:  
 - Initializes the Spark session, logger, and file paths.  
 - Configures the `ETLTestCases` class with the required dependencies.  
 - Executes the test cases using `unittest.TextTestRunner`.

### Spark Session Initialization

Purpose:  
 - Sets up a local Spark session with the application name "ETLTest".  
  
 Implementation:  
 - Uses `SparkSession.builder` to configure the session.  
 - Sets the master to "local[2]" to use two threads for local execution.

### Logger Setup

Purpose:  
 - Configures a logger for the testing process to capture important events and errors.  
  
 Implementation:  
 - Uses Python's `logging` module to create a logger named "ETLTestLogger".  
 - Configures the logger with a stream handler and a formatter for readable logs.

### File Paths and Configuration

Purpose:  
 - Maps logical table names to their corresponding sample data files (CSV format).  
 - Injects the configuration and file paths into the `ETLTestCases` class for testing.

### Test Execution

Purpose:  
 - Executes the test cases defined in `ETLTestCases`.  
  
 Steps:  
 1. Loads all test cases using `unittest.TestLoader`.  
 2. Runs the test cases using `unittest.TextTestRunner`.  
 3. Stops the Spark session after the tests are completed.