1 a)

* The template method has been used to abstract out similarities between the implementation of MaxValueCalculation and MinValueCalculation
* The Singleton pattern was also used with the LargeHadronCollider where there is only one static instance of the class that is obtained with getInstance().
* The strategy pattern is being used to implement different kinds of Loggers

HU Edit: I disagree, the specific implementation of Logger to use is chosen in the constructor. The strategy pattern would have a Logger as a paramater to the constructor, allowing a LargeDataSetCalculation to be created with the logger of choice.

* The decorator pattern is being used to a FileSystemLogger with a CvsFormattedLogger

b)

1. The constructor uses a very specific instance of a Logger (FileSystemLogger) with a very specific file path “/tmp/log.csv”. This can cause problems as the user is unable to reuse the calculations class with a different Logger as the is tight coupling between the class and the FileSystemLogger
2. The data field should not be protected as it contains sensitive “data” that should not be part of the class’ API. If a class extends LargeDataSetCalculation it has full access to the data field which should be limited.
3. The use of the Singleton object LargeHadronCollider.getInstance() is bad as it creates tight coupling between the two classes. Other sets of data can not be used :(

HU Edit: The Singleton pattern is not the problem here. There is only one Large Hadron Collider so it makes sense. The LHC class probably contains more than just the data. But yes, I agree, other data sets should be usable.

1. Min and Max class doCalculation() functions have a lot of code duplication
2. Unclear that LargeDataSetCalculation will output a csv at a specified filepath
3. Timestamp tightly coupled with LargeDataSetCalculation despite not relevant to the class itself
4. Min/MaxValueCalculation are tightly coupled with LargeDataSetCalculation so cannot be reused other scenarios.

c)

1. This can be resolved by using a composite object. The constructor should take in as a parameter an object that implements the Logger interface. The object should then be stored in the logger field.
2. Make the data field private and pass to doCalculation.
3. Do not use Singleton object. Instead have the 2D array inserted into the function through a parameter (could use a singleton object, but in this case would need to pass it in to the constructor to be able to switch it out in case want to use another type of singleton object. HU: Singleton pattern not the problem here. That’s about the implementation of the LHC class, not the coupling. Better to use the strategy pattern, where the data set, in this case the LHC is a ‘strategy’. Introduce interface DataSet with method retrieveData(). Give LargeDataSetCalculation a field ‘private DataSet dataSet;’ and set it in the constructor (which now takes a DataSet as a parameter). Then the first line of calculate() becomes:
   * data = dataSet.retrieveData();
4. Extract and move the duplicated code to superclass, create and store the code in a new function to be called in subclass. Alternatively, have one ‘FlatReduceCalculation’ class which takes a BiFunction<Integer, Integer, Integer>, and uses that function to reduce the 2D input array into a singular value. Then provide two default factory style static functions .minimum() and .maximum() that automatically populate that BiFunction.
5. Use a factory method for construction of LargeDataSetCalculation e.g. `.newCSVResult(String path)`
6. Decorate the provided Logger with a new class TimestampedLogger which will automatically add the timestamp and provide the functionality for that internally.
7. Move Min/MaxValueCalculation to become a strategy pattern as opposed to Template pattern.