***Instructions***

* I have included the database .bak file that I used as well as the separate sql scripts for the stored procedures I used and table creation.
* The ISO 4217 codes were imported into database so there is no script to populate the data.
* I included both in case one doesn’t work or not.
* There is also a sample spreadsheet that I used to do some preliminary testing.
* I have included logging so within web.config it set for C:\temp folder.
* I have used Visual Studio 2015

***Notes***

* I didn’t get time to implement proper exception handling or finish off the unit tests and logging.
* I tried to adhere to the solid principles as much as possible and used various design patterns such as repository when required.
* I choose Dapper has the micro ORM purely as performance was a key consideration in the design of this application.
* I did use some aspects of **Domain driven design**, in terms of structuring of my code.
* Unfortunately I didn’t get the last part referring to the total to be correct when there are errors (it works fine without any errors), due to how the threads are run in parallel. I know that having a counter in **Parrallel.For or ForEach** will not work properly unless locking is implemented which then can negate the performance improvement gained from using it in the first place. Hence why I used the **Interlocked.Increment** method to create the atomic operation and preventing any race conditions. However that didn’t mitigate the issue, and I can’t see what the resolution would be currently. I doubt very much by using **MaxDegreeOfParrallelism** would help.
* I was thinking of using **SqlBulkCopy** but didn’t get time to try it out as an alternative. I know this would definitely be good for performance, but not sure which would be faster without benchmarking.
* I enjoyed this task, and got to learn more about multithreading programming. I certainly think, one of the key aspects in development is to always keep learning and assimilating information.