Batch Job considerations

1. Iterable interface
   1. Apply heap limits when loading the data through CSV file and construct objects
   2. Apply SOQL governor limits, it cannot query more than 50000
2. Stateful interface
   1. We can track the failed records and re-initiate the batch with failed records
   2. We can skip certain records to process later (Ex: If I want to insert record B after insertion of record A)
   3. It leads to heap size limit in the case of file uploads or large amount of data
3. Execute method
   1. Execute method does not run immediately. It runs only when resources are available.

There will be time lag between start method and execute method execution.

In between there might be a chance the records will be update by some other process or from user interface itself.

1. Start method
   1. Don’t use subqueries in start method to query the data, it gives CPU timeout error
2. Locking
   1. Need to optimize the batch job size to avoid timeout limits
   2. Need to process the batch jobs in serial mode instead of parallel mode
   3. Sometimes, we need to turn off the triggers, workflows, process builders to avoid dead lock situations
   4. If any job takes long time, it will give impact on other jobs to delay
   5. Sometimes, we need to block the users not to validate through UI to avoid interrupted results and avoid dead locks in the case of master detail relationships.
3. It is not real time data.
4. Order of execution is not guaranteed.
   1. need to call order service callout before calling payment service
   2. need to insert/update parent records first and then child records
5. scheduled the batch jobs
   1. failed callouts
   2. failed jobs due to complex computations