Whenever we try to work with batch Apex there are a couple of things that we need to keep in mind.

We are not supposed to write all the complex code that we write in Apex classes within batch apex.

We will be looking at a couple of bottleneck that we will be coming across when we try to work with inner queries, batch Apex and iterators, post which will walk you through the workarounds.

First, when we try to put inner query within the start method, the batch will start to show unexpected behaviour.

Here is the sample code!

```
public Database.QueryLocator start(Database.BatchableContext context) {
    String query = 'SELECT Id,
                          (SELECT Id FROM Contacts),
                          (SELECT Id FROM Opportunities) FROM Account';
    return Database.getQueryLocator(query);
}
```

```
public void execute(Database.BatchableContext context, List<Account> scope) {
    // do something
}
```

Workaround for this is to have the SOQL query without inner queries in the start method and then in the execute method we need to re-query the sObject with inner queries.

Here is the refactored code.

```
public Database.QueryLocator start(Database.BatchableContext context) {
    String query = 'SELECT Id FROM Account';
    return Database.getQueryLocator(query);
}
```

```
public void execute(Database.BatchableContext context, Account[] scope) {
  Account[] records = [SELECT Name,
                              (SELECT Name FROM Contacts),
                              (SELECT Name FROM Opportunity)
                              FROM Account WHERE Id =: scope];
  // do something
}
```

Second, we are not supposed to use iterator() method multiple times in the start method, it gives us back unexpected results.

Here is the code!

```
Database.QueryLocator queryLocator = Database.getQueryLocator([SELECT Id FROM Account]);

if(!queryLocator.iterator().hasNext())
    //do something

if(!queryLocator.iterator().hasNext())
    //do something
```

As per the documentation

> To iterate over a query locator, save the iterator instance that this method returns in a variable and then use this variable to iterate over the collection. Calling iterator every time you want to perform an iteration can result in incorrect behaviour because each call returns a new iterator instance.

So we need to call `iterator()` once per `QueryLocator` and we can call the `hasNext()` method multiple times on the iterator instance multiple times.

Here is the refactored code.

```
Database.QueryLocator queryLocator = Database.getQueryLocator([SELECT Id FROM Account]);
Database.QueryLocatorIterator it = queryLocator.iterator();
if(!it.hasNext())
    //do something
```

```
if(!it.hasNext())
    //do something
```

If you were to iterate over the result set multiple times then try to save each record in to a list and you are free to iterate the list.