How to avoid "Too many query rows: 50001"

If we query all account records without a where clause and process the records with New York as the billing state in a for loop. We would encounter the Too Many Query Rows 50001 error. However if we add a where clause with BillingState = 'NY', the query is now more restrictive by using a where clause which will help reduce the amount of records returned.

```
//Open ended SOQL query
List<Account> accounts = [SELECT id, Name FROM Account];

//More restrictive SOQL query
List<Account> accountsInNY = [SELECT id, Name FROM Account WHERE BillingState = 'NY'];
```

COPY CODE

Using `batch` apex is another way to avoid encountering the "To many query rows 50001" error.A batch apex job for the account object can return a QueryLocator for all account records, up to 50 million records in an organization.

Every execution of a batch apex job is considered a discrete transaction.When a batch is executed without the optional scope parameter from database.executeBatch it is split into apex transactions of 200 records. Apex governor limits are reset for each transaction. Important to also note: if the first apex transaction succeeds but the second one fails. The first transaction is `not` rolled back.

```
global class AccountBatch implements
    Database.Batchable<sObject>, Database.Stateful {

    global Database.QueryLocator start(Database.BatchableContext bc) {
        //Query the records - return accounts with NY billing state
        return Database.getQueryLocator(
            'SELECT ID, BillingStreet, BillingCity, BillingState, ' +
            'BillingPostalCode FROM Account ' +
            'Where BillingState = \'NY\''
        );
    }
    global void execute(Database.BatchableContext bc, List<Account> scope){
        // process each batch of records
        for(Account account:scope){
            //Process account records with NY as billing state
        }

        update scope;
    }
    global void finish(Database.BatchableContext bc){
        // execute any post-processing operations that are needed
    }
}
```

COPY CODE

Now that we are using batch apex - the records will be processed in batches of 200 records.

We can run the batch apex from anonymous apex with the following code.

```
AccountBatch myBatchObject = new AccountBatch();
Id batchId = Database.executeBatch(myBatchObject);
```