**Query Error: "System.QueryException: Non-selective query against large object type"**

To avoid "Query Error: "System.QueryException: Non-selective query against large object type"" exception, follow the below steps  
  
1. Make sure that your SOQL doesn't have null values in set or list or in a variable while fetching. i.e The filter value includes null (for instance binding with a list that contains null)

Note: filter out all null records in your Apex query (where fieldname\_\_c!=null).

V.V.V imp:  Right operand of where clause should not be null or blank, for example: **where Name in :setOfNames;** and setOfNames must be initialized or should not be blank.

V.V.V. imp: Avoid comparing with an empty value (Name != ”).

2. Don't use negative operator such as NOT EQUAL TO (or !=), NOT CONTAINS, and NOT STARTS WITH in SOQL to fetch records.  
  
3. Avoid using Formula field for filtering in SOQL.( Do not use a formula field for filtering in SOQL.)

if you have a formula field that you are using as a unique key to prevent duplicates, eventually you are going to hit this problem. The only fix I have found is to try and have a condition that reduces the number of records the query has to look at.  
  
4. If you use IN operator, make sure that the field is indexed.  
  
5. If a field is not indexed, kindly contact Salesforce.com to do custom index on the field.  
  
6. Empty your Organization Recycle Bin.

7.  It’s best to try to have that field indexed by checking the “external ID” checkbox when you create that field.  Also, as a best practice, always use WHERE clause in your SOQL statements to read the records.

8. Data skew exists whereby the number of matching rows is very large (for instance, filtering for a particular foreign key value that occurs many times)

9. Avoid using OR conditions in SOQL on objects having more than 200,000 records. Prefer using AND condition, but if need to use it you may use that in loop using If conditions.

**Note:**  
  
1. By default, Lookup, Master Detail and external id fields are indexed.  
  
2. For custom indexing, kindly contact Salesforce.com support.  
  
Check this link for more info  
  
<https://www.salesforce.com/us/developer/docs/apexcode/Content/langCon_apex_SOQL_VLSQ.htm>

## **Few More ROOT CAUSES OF A NON-SELECTIVE SOQL QUERY?**

* **Too many records** - again, when you get a table above 100k rows and start asking for records that meet the condition of all 100k rows you're going to hit this issue. For example, let's say all your Contacts were imported on Aug 1, 2011 when you migrated to Salesforce. If you run a query that says:

**SELECT id FROM Contact WHERE CreatedDate  > 2010-01-01T00:00:00Z**  
 THIS IS GOING TO RETURN ALL YOUR CONTACTS

* **Using the good ol' Wildcard -**a LIKE condition that has a leading "%" WILL NOT USE an index

**SELECT id FROM Account WHERE Name LIKE ‘%Acme%’** //this would be better suited for a SOSL query

         WHen you build a report or use a list view where you add a Contains clause that is the same

         as doing what we have adove - it essentially translates to %Value%.

* **Using NOT and != in your query** - When you filter on these two [inequality filters](https://developer.salesforce.com/blogs/engineering/2013/05/dealing-with-exception-filters-in-force-com.html) the query optimizer cannot take advantage of the index to make the query faster. By doing the opposite and filters on "=" or using an IN ('a','b','c').
* Complex Join statements - when you start to build in complex AND/OR logic into the queries and adding sub-queires - this forces the query optimizer to produce and optimized query for the join. In reality it may be a lot more efficient to break these into multiple queries. I have found this to be particularly true when I see a lot of OR logic in the queries I look at.

If Salesforce is going to use an index when an OR condition is involved - ALL the fields you have listed in the OR condition must be indexed fields.

Note: if we try to break the large query into smaller queries. that would cause governor limit problems.

Best practice-1: Before putting the query in apex, make sure SOQL is not breaking governer limit. One may check governer limit of SOQL by using a simple debug in apex System.debug('LimitOfSOQLRows:'+Limits.getLimitQueryRows());

Best Practice-2: Put your query in Query plan tool of Salesforce Developer console to ensure that it is  selective and ready to be used in apex.

The platform automatically maintains indexes on the following fields for most objects.

* RecordTypeId
* Division
* CreatedDate
* Systemmodstamp (LastModifiedDate)
* Name
* Email (for contacts and leads)
* Foreign key relationships (lookups and master-detail)
* The unique Salesforce record ID, which is the primary key for each object
* Salesforce also supports custom indexes on custom fields, with the exception of multi-select picklists, text area (long), text area (rich), non-deterministic formula fields, and encrypted text fields.

External IDs cause an index to be created on that field, which is then considered by the Force.com query optimizer. External IDs can be created on only the following fields.

* Auto Number
* Email
* Number
* Text
* To create custom indexes for other field types, including standard fields, contact salesforce.com Customer Support

<https://developer.salesforce.com/forums/?id=906F00000008sbTIAQ>