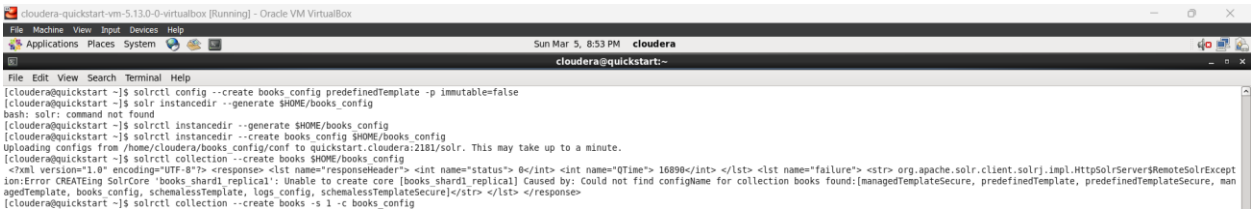
Assignment 5

Big Data and Data Science

Task 1: Dealing Book Data

Creating the configs and collection



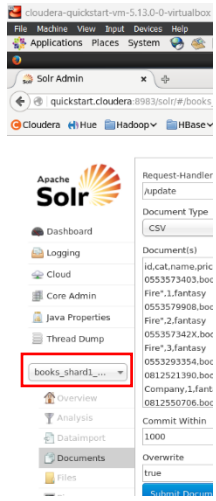
```
[cloudera@quickstart ~]$ solrctl config --create books_config predefinedTemplate -p immutable=false
[cloudera@quickstart ~]$ solr instancedir --generate $HOME/books_config
bash: solr: command not found
[cloudera@quickstart ~]$ solrctl instancedir --generate $HOME/books_config
[cloudera@quickstart ~]$ solrctl instancedir --create books_config $HOME/books_config
Uploading configs from /home/cloudera/books_config/conf to quickstart.cloudera:2181/solr. This may take up to a minute.
[cloudera@quickstart ~]$ solrctl collection --create books $HOME/books_config
 <?xml version="1.0" encoding="UTF-8"?> <response> <lst name="responseHeader"> <int name="status"> 0</int> <int name="QTime"> 16890</int> </lst> <lst name="failure"> <str> org.apache.solr.client.solrj.impl.HttpSolrServer$RemoteSolrExcept
ion:Error CREATEing SolrCore 'books_shard1_replica1': Unable to create core [books_shard1_replica1] Caused by: Could not find configName for collection books found:[managedTemplateSecure, predefinedTemplate, predefinedTemplateSecure, man
agedTemplate, books_config, schemalessTemplate, logs_config, schemalessTemplateSecure]</str> </lst> </response>
[cloudera@quickstart ~]$ solrctl collection --create books -s 1 -c books_config
```

There are a series of steps involved before creating the collection and loading the data. First create a config file for books by the name of books_config and then create a path for the config file by using generate command followed by path. Now we are good to create a collection. Now create a collection with name books using create command as shown in the above figure. All the commands in the above figure needs to be executed in the same order.

Schema.xml for books data set

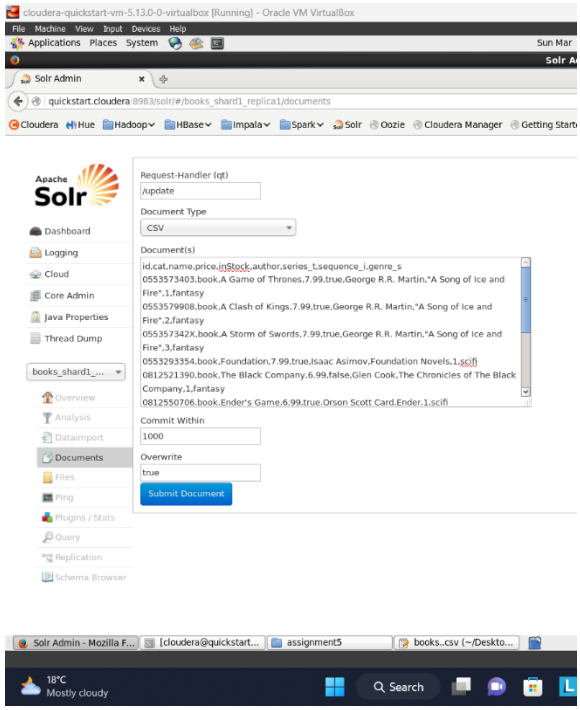
```
<!-- Common metadata fields, named specifically to match up with
  SolrCell metadata when parsing rich documents such as Word, PDF.
  Some fields are multiValued only because Tika currently may return
  multiple values for them. Some metadata is parsed from the documents,
  but there are some which come from the client context:
    "content_type": From the HTTP headers of incoming stream
    "resourcename": From SolrCell request param resource.name
-->
<field name="title" type="text_general" indexed="true" stored="true" multiValued="true"/>
<field name="subject" type="text_general" indexed="true" stored="true"/>
<field name="description" type="text_general" indexed="true" stored="true"/>
<field name="comments" type="text_general" indexed="true" stored="true"/>
<field name="author" type="text_general" indexed="true" stored="true"/>
<field name="keywords" type="text_general" indexed="true" stored="true"/>
<field name="category" type="text_general" indexed="true" stored="true"/>
<field name="resourcename" type="text_general" indexed="true" stored="true"/>
<field name="url" type="text_general" indexed="true" stored="true"/>
<field name="content_type" type="string" indexed="true" stored="true" multiValued="true"/>
<field name="last_modified" type="date" indexed="true" stored="true"/>
<field name="links" type="string" indexed="true" stored="true" multiValued="true"/>

<!-- Main body of document extracted by SolrCell.
     NOTE: This field is not indexed by default, since it is also copied to "text"
     using copyField below. This is to save space. Use this field for returning and
     highlighting document content. Use the "text" field to search the content. -->
<field name="content" type="text_general" indexed="false" stored="true" multiValued="true"/>


<!-- catchall field, containing all other searchable text fields (implemented
     via copyField further on in this schema  -->
<field name="text" type="text_general" indexed="true" stored="false" multiValued="true"/>

<!-- catchall text field that indexes tokens both normally and in reverse for efficient
     leading wildcard queries. -->
<field name="text_rev" type="text_general_rev" indexed="true" stored="false" multiValued="true"/>

<!-- non-tokenized version of manufacturer to make it easier to sort or group
     results by manufacturer.  copied from "manu" via copyField -->
<field name="manu_exact" type="string" indexed="true" stored="false"/>

<field name="payloads" type="payloads" indexed="true" stored="true"/>

<field name="_version_" type="long" indexed="true" stored="true"/>
```

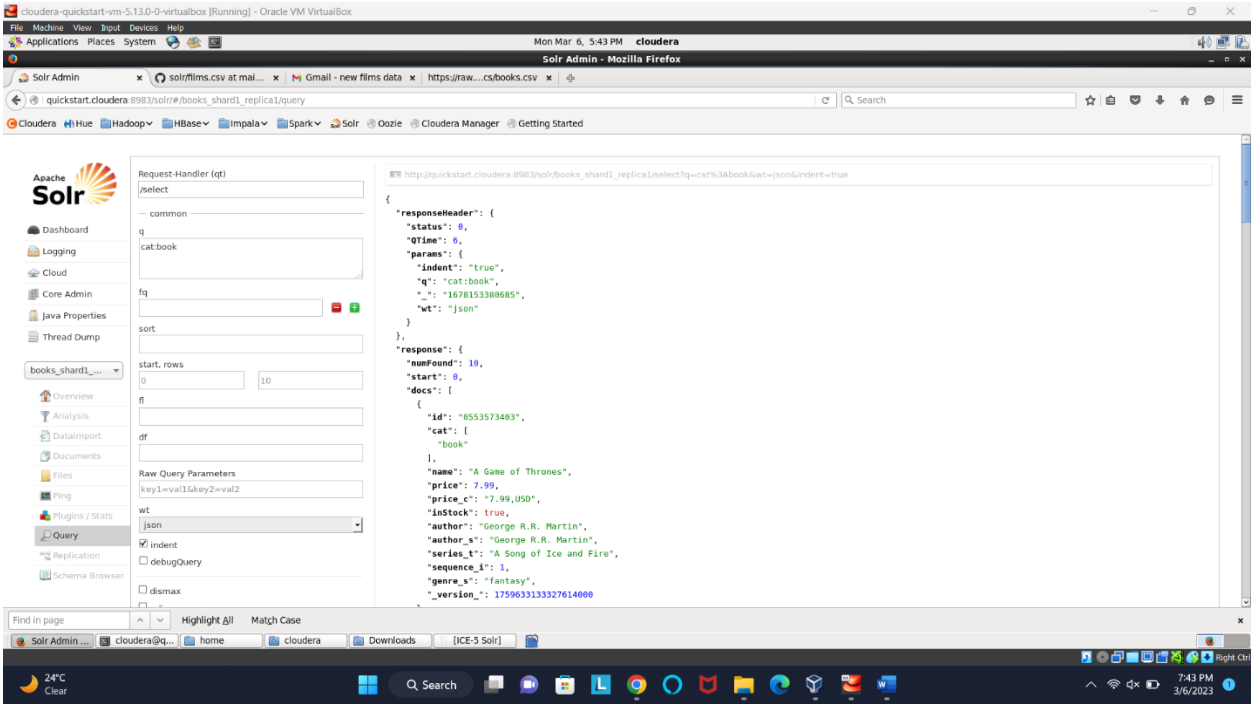
Then open solr in browser in cloudera. Solr will run on localhost:8983 by default. On successful execution of above commands a new core can be seen in the side navigation bar.



Click on document under the core and add the csv data that is obtained from the github link in the canvas assignment document.

Select the data type as csv and click on submit document and the will get added successfully.

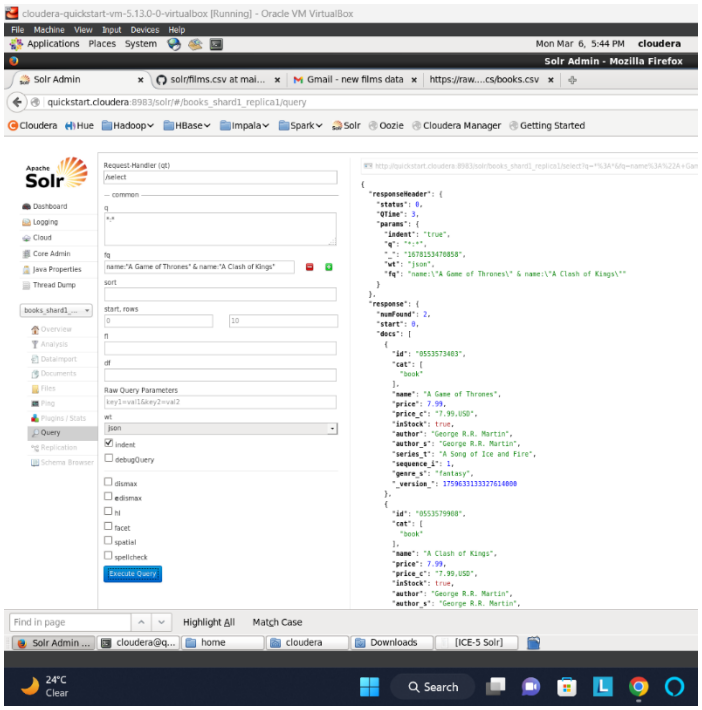




A: Display all the books with its attributes

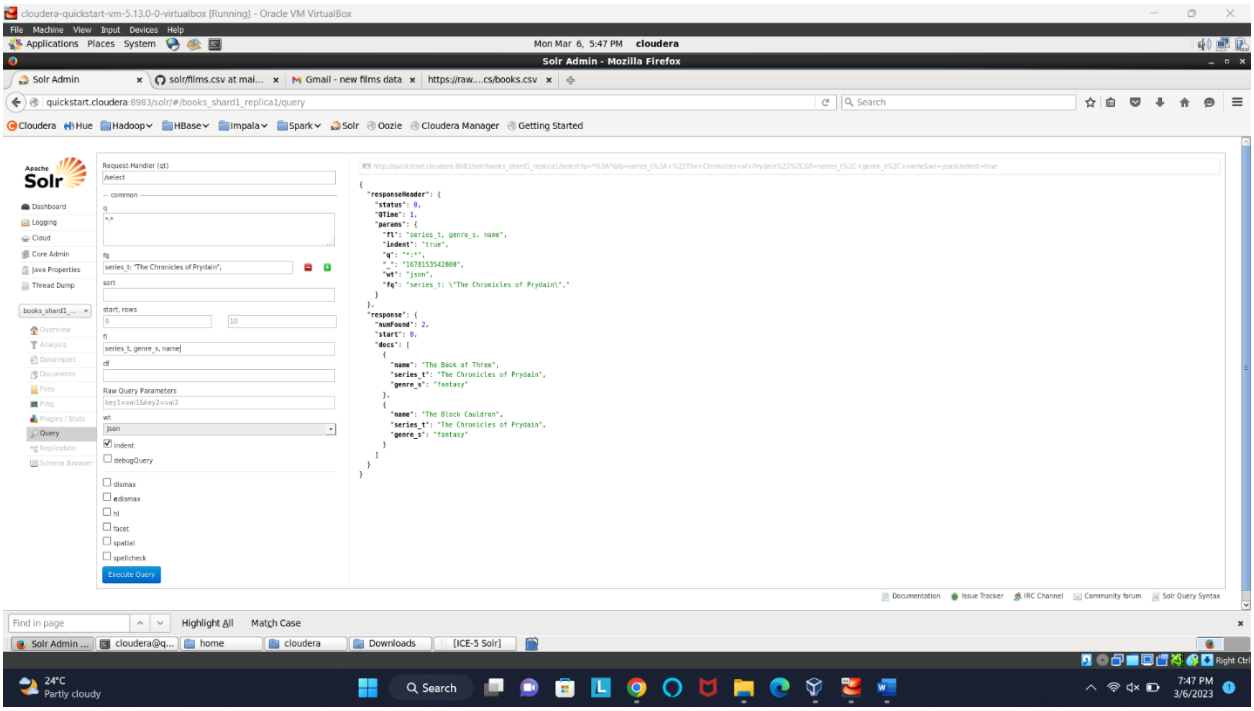
With cat:book in q, on hit of the execute query button, the result of the query is available in json format on the right side of the screen.

B: To obtain the data whose book name is game of thrones and a clash of kings with their author's name.



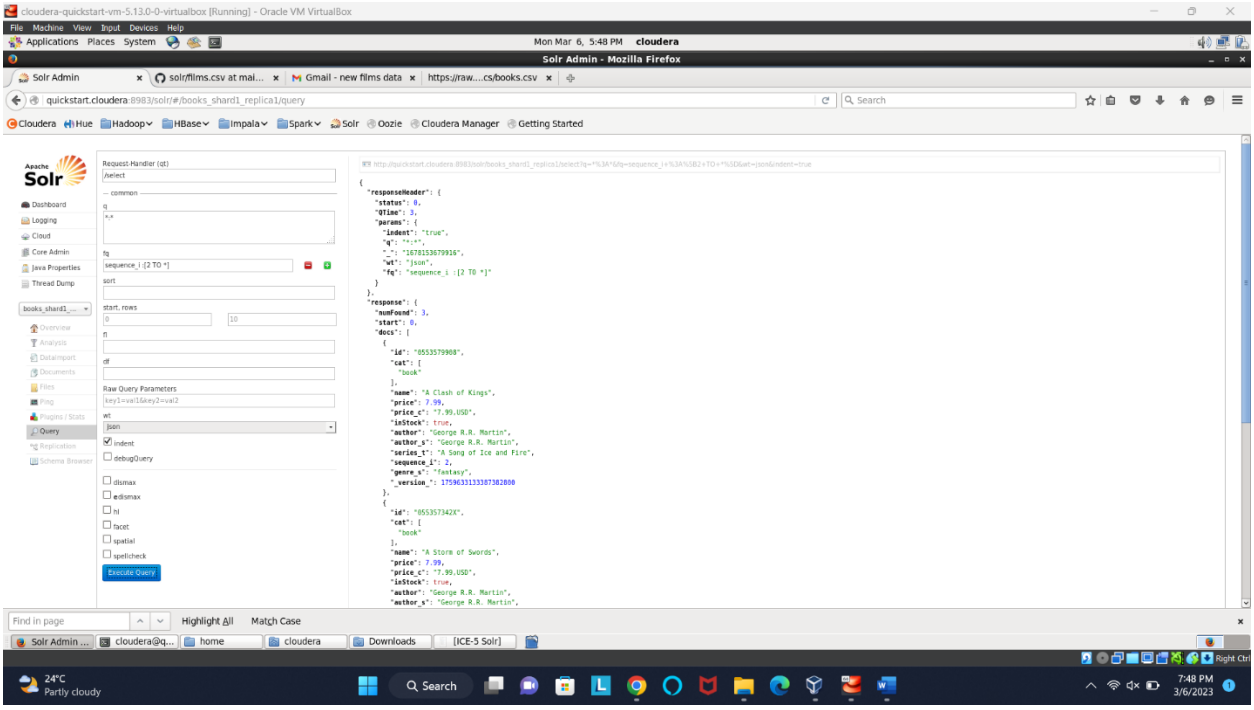
Query with the name of the movie separated by and symbol and name of the other movie in fq text box. Then click on execute query. The result is the data which has the name of the movie specified in fq column.

C: Get all the books which has series_t as chronicle of prydain with their genere.



Add the condition in fq text box with series_t:<name> and add the required attribtes in the output to fl column. On click of the execte button the result in json will be available in the right side of the screen.

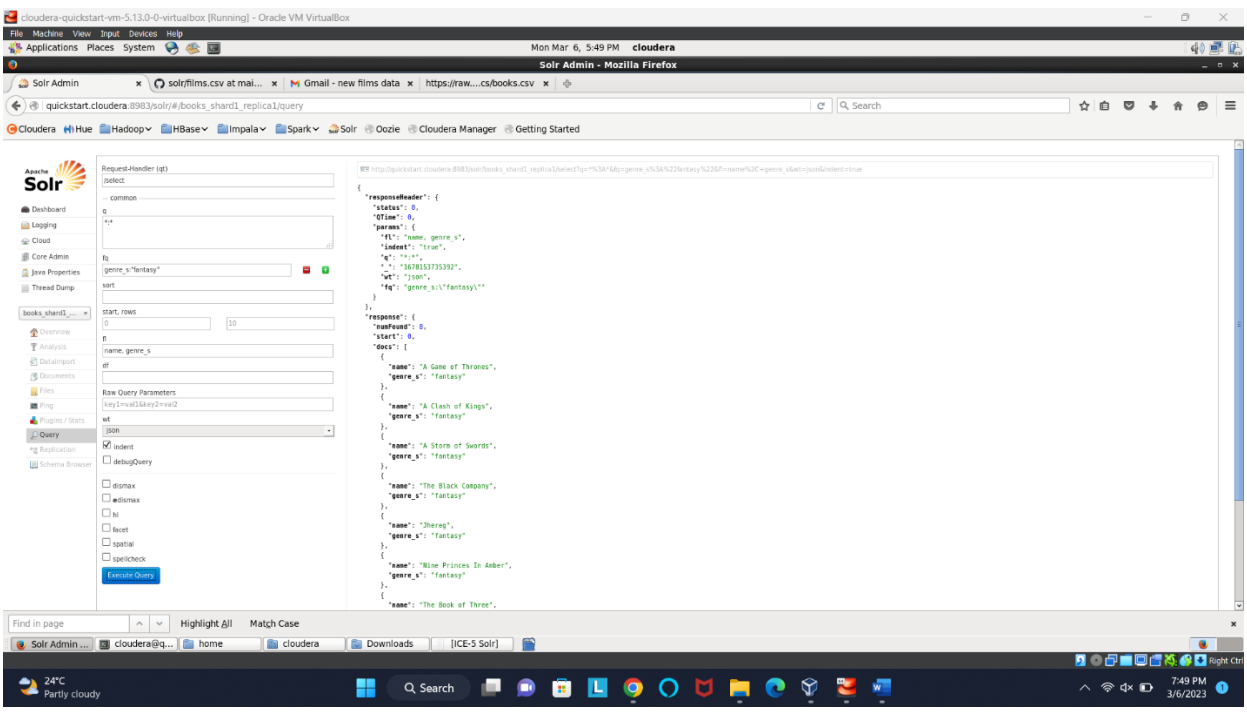
D: get all the records which has sequence greater than 1.



To achieve this task, the condition for sequence greater than 1 can be written in fq as sequence:[2 TO *]. Here 2 is inclusive and * means the maximum that is available in the data. On click of the execute button the data whose sequence is greater than 1 is available on the screen.

E: finding books of fantasy genre.

This can be achieved by adding the condition genre=Fantasy in the fq column. The result can be limited to few attributes by adding the names of desired attributes in the fl field. On execution the query the result will be available on the screen with the books that are of genre fantasy.



Task 2

Creating config file and collection.

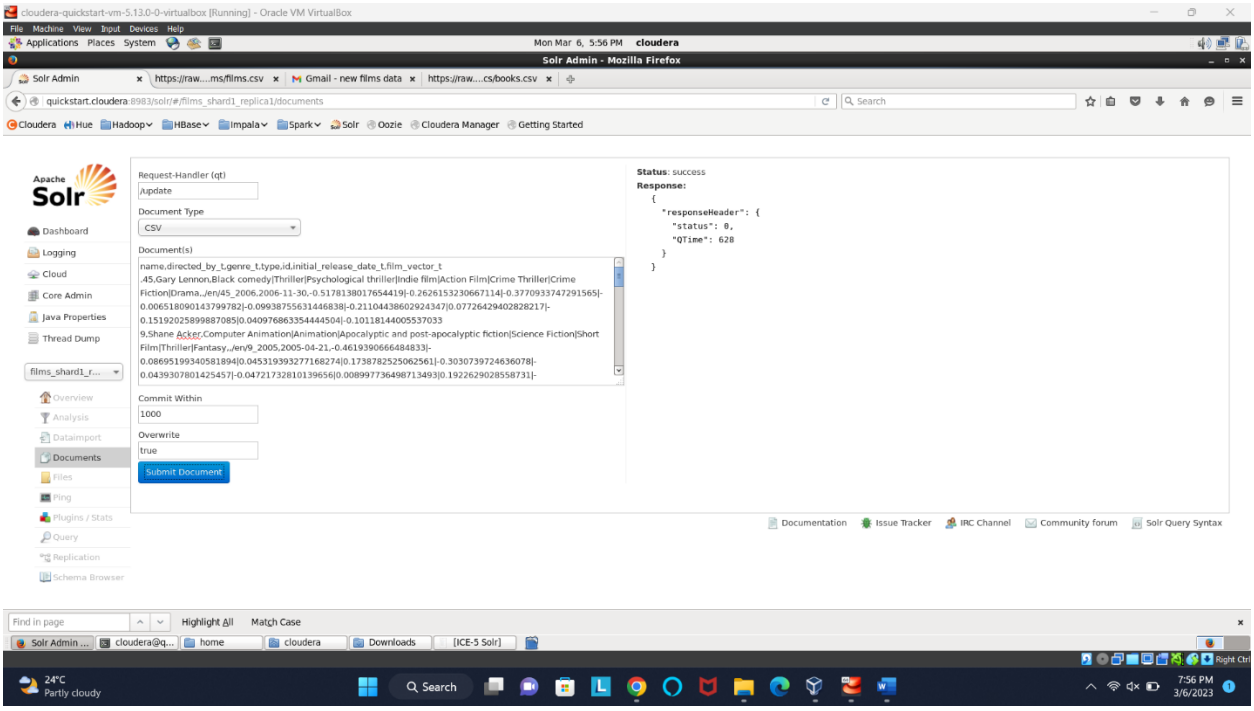
```
[cloudera@quickstart ~]$ solrctl config --create books_config predefinedTemplate -p immutable=false
[cloudera@quickstart ~]$ solr instancedir --generate $HOME/books_config
bash: solr: command not found
[cloudera@quickstart ~]$ solrctl instancedir --generate $HOME/books_config
[cloudera@quickstart ~]$ solrctl instancedir --create books_config $HOME/books_config
Uploading configs from /home/cloudera/books_config/conf to quickstart.cloudera:2181/solr. This may take up to a minute.
[cloudera@quickstart ~]$ solrctl collection --create books $HOME/books_config
 <?xml version="1.0" encoding="UTF-8"?> <response> <lst name="responseHeader"> <int name="status"> 0</int> <int name="QTime"> 16890</int> </lst> <lst name="failure"> <str> org.apache.solr.client.solrj.impl.HttpSolrServer$RemoteSolrExcept
ion:Error CREATEing SolrCore 'books_shard1_replica1': Unable to create core [books_shard1_replica1] Caused by: Could not find configName for collection books found:[managedTemplateSecure, predefinedTemplate, predefinedTemplateSecure, man
agedTemplate, books_config, schemalessTemplate, logs_config, schemalessTemplateSecure]</str> </lst> </response>
[cloudera@quickstart ~]$ solrctl collection --create books -s 1 -c books_config
[cloudera@quickstart ~]$ solrctl config --create films_config predefinedTemplate -p immutable=false
[cloudera@quickstart ~]$ solrctl instancedir --generate $HOME/films_config
[cloudera@quickstart ~]$ solrctl instancedir --create films_config $HOME/films_config
Uploading configs from /home/cloudera/films_config/conf to quickstart.cloudera:2181/solr. This may take up to a minute.
[cloudera@quickstart ~]$ solrctl collection --create films -s 1 -c films_config
[cloudera@quickstart ~]$
```

This is similar to the creation of books collection. Create the config and collections by executing the commands shown in the above figure.

Schema.xml for films dataset

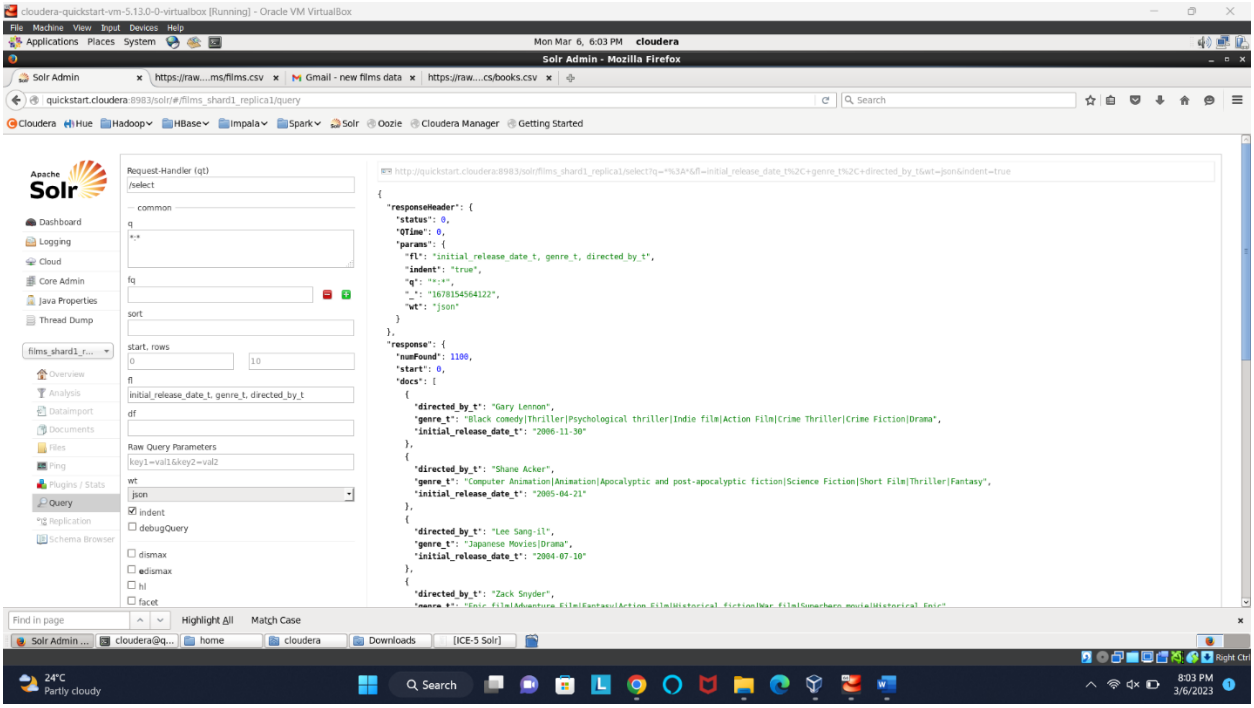
```
<!-- Common metadata fields, named specifically to match up with
  SolrCell metadata when parsing rich documents such as Word, PDF.
  Some fields are multiValued only because Tika currently may return
  multiple values for them. Some metadata is parsed from the documents,
  but there are some which come from the client context:
    "content_type": From the HTTP headers of incoming stream
    "resourcename": From SolrCell request param resource.name
-->
<field name="title" type="text_general" indexed="true" stored="true" multiValued="true"/>
<field name="subject" type="text_general" indexed="true" stored="true"/>
<field name="description" type="text_general" indexed="true" stored="true"/>
<field name="comments" type="text_general" indexed="true" stored="true"/>
<field name="author" type="text_general" indexed="true" stored="true"/>
<field name="keywords" type="text_general" indexed="true" stored="true"/>
<field name="category" type="text_general" indexed="true" stored="true"/>
<field name="resourcename" type="text_general" indexed="true" stored="true"/>
<field name="url" type="text_general" indexed="true" stored="true"/>
<field name="content_type" type="string" indexed="true" stored="true" multiValued="true"/>
<field name="last_modified" type="date" indexed="true" stored="true"/>
<field name="links" type="string" indexed="true" stored="true" multiValued="true"/>

<!-- Main body of document extracted by SolrCell.
     NOTE: This field is not indexed by default, since it is also copied to "text"
     using copyField below. This is to save space. Use this field for returning and
     highlighting document content. Use the "text" field to search the content. -->
<field name="content" type="text_general" indexed="false" stored="true" multiValued="true"/>


<!-- catchall field, containing all other searchable text fields (implemented
     via copyField further on in this schema  -->
<field name="text" type="text_general" indexed="true" stored="false" multiValued="true"/>

<!-- catchall text field that indexes tokens both normally and in reverse for efficient
     leading wildcard queries. -->
<field name="text_rev" type="text_general_rev" indexed="true" stored="false" multiValued="true"/>

<!-- non-tokenized version of manufacturer to make it easier to sort or group
     results by manufacturer.  copied from "manu" via copyField -->
<field name="manu_exact" type="string" indexed="true" stored="false"/>

<field name="payloads" type="payloads" indexed="true" stored="true"/>

<field name="_version_" type="long" indexed="true" stored="true"/>
```

Then, open solr in the browser and the new collection will be available to use. Select the films core and click on documents and add the csv file in the document field. Increase the commit width to 10000 as the file is huge it will take some extra time.

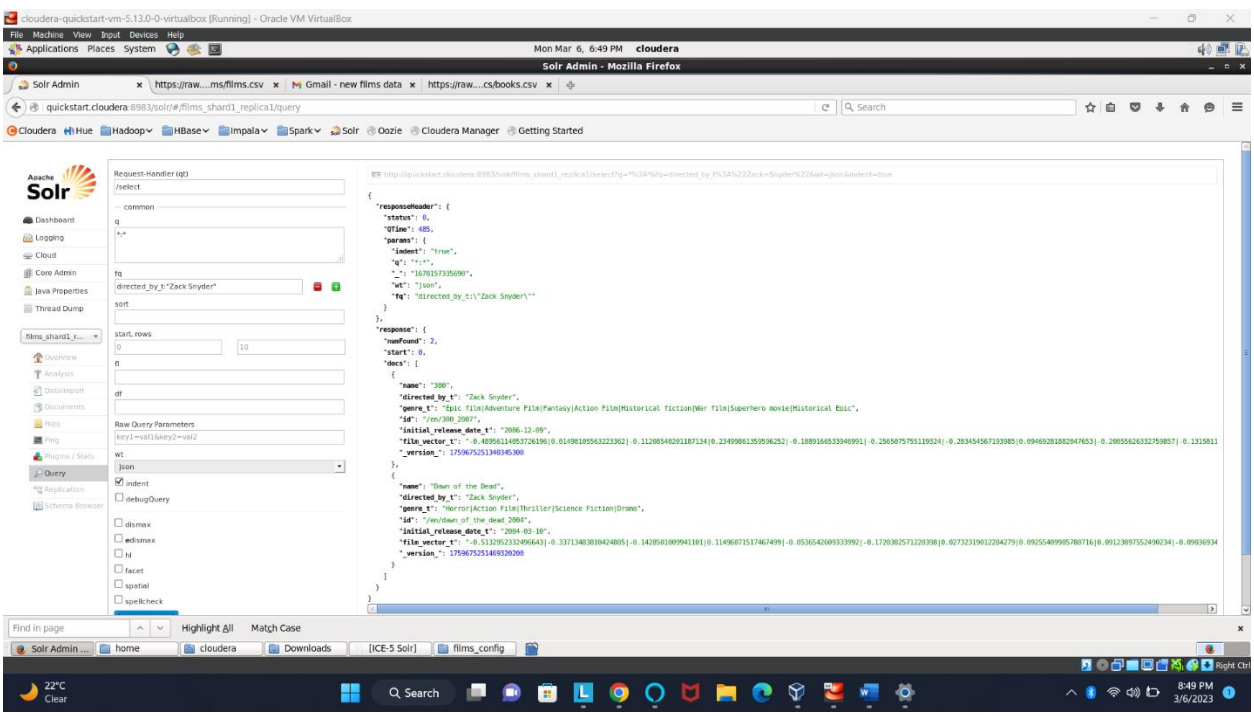


A: Fetch all the movies with the attributes initial release date, genre and directed by.

Add the desired column names in the fl column click on the execute query button. The result in JSON format is available on the screen with the specified fields only.

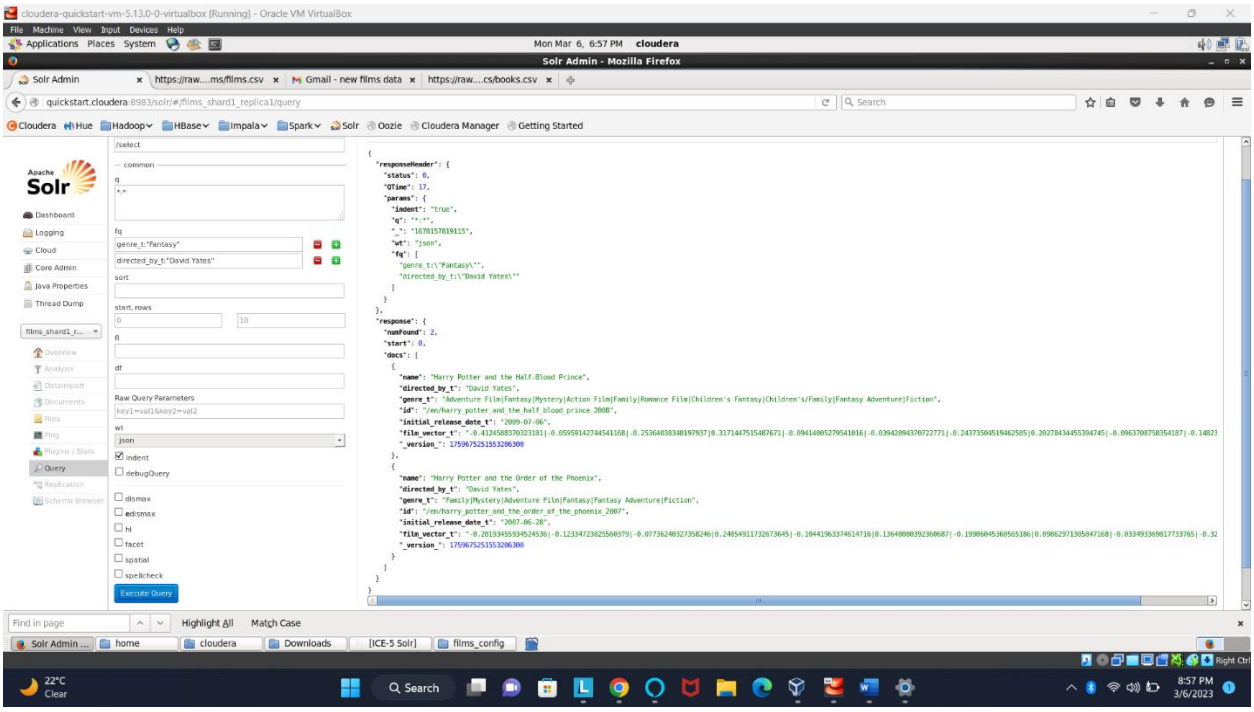


B) Obtain all the films which are directed by Zack Snyder



This can be obtained by adding the condition for directed_by_t column with the name of the director Zack Snyder.

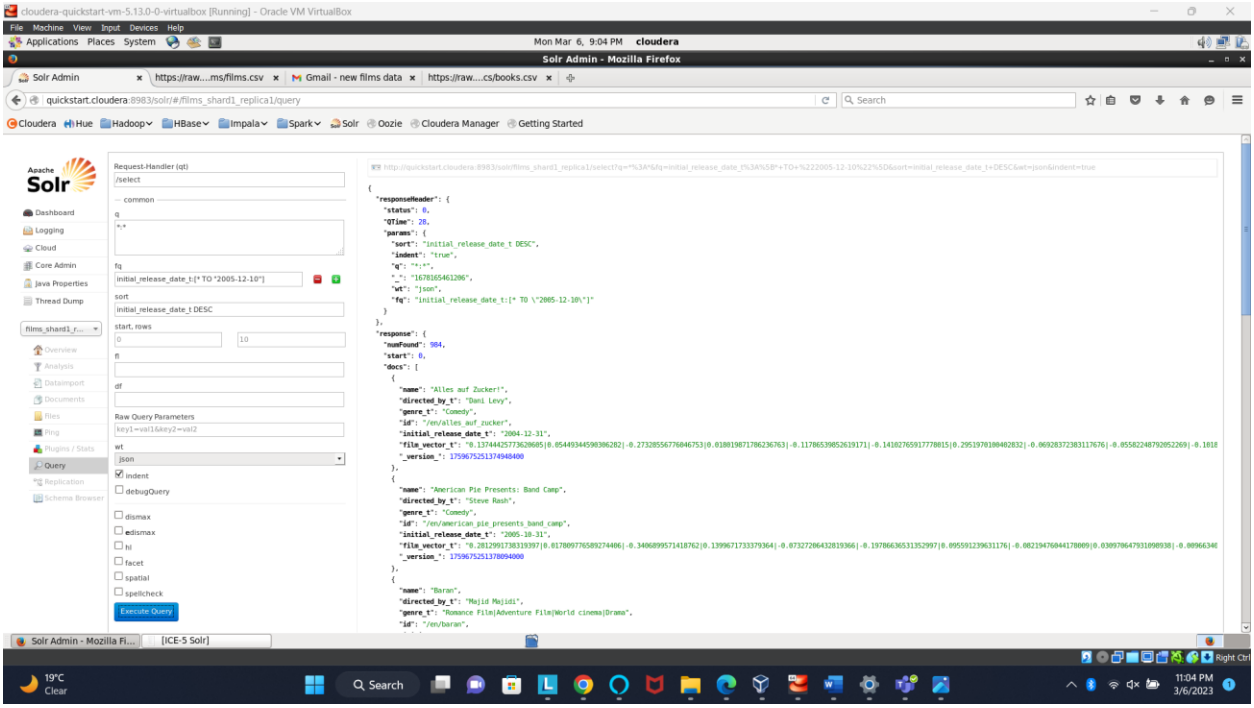
C) get all the films whose director is David Yates and the films are of genre fantasy.



Add the condition genre as fantasy in fq field and click on the plus icon to add a new condition. Add the new condition as directed by David yates in the newly added fq field. Execute the query and the result will be right on the screen.

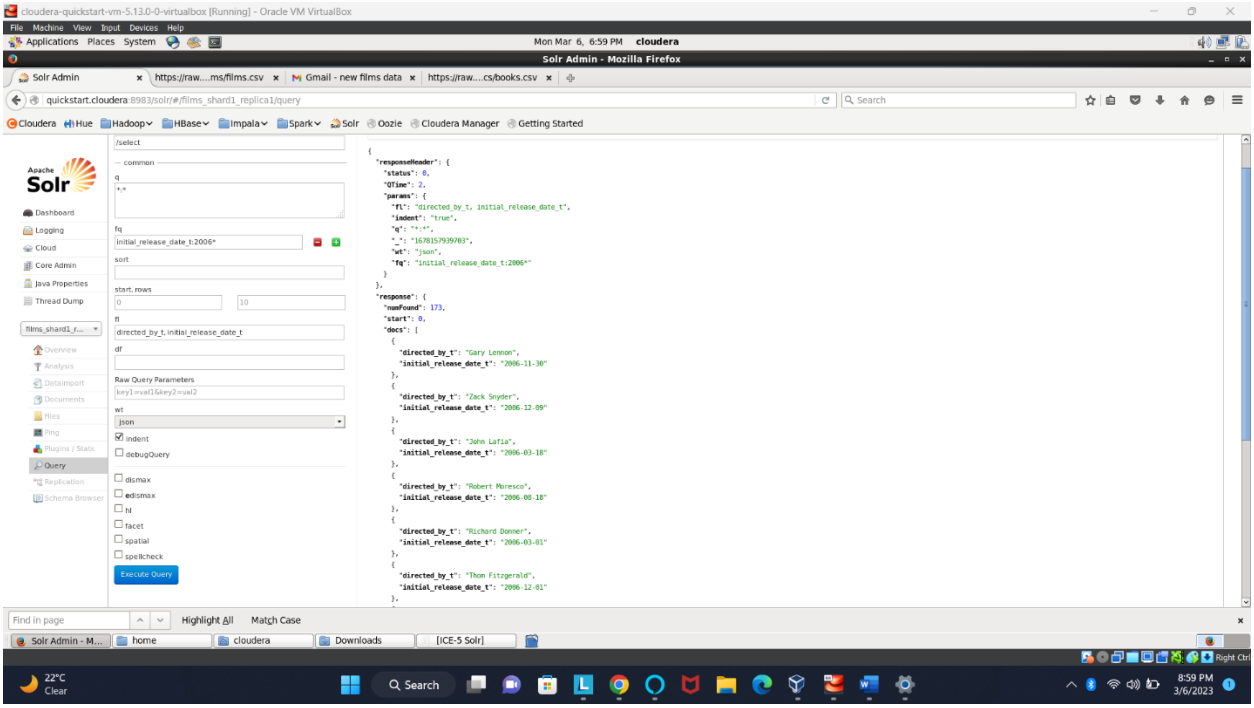
D) Obtain all the films which have the release date before December 10th 2005 and sort the date of release in descending order.

Add the condition in the fq filed in the solr page and add the value initial_release_date_t desc in the sort field.



E) Give all the directors whose films were released in the year 2006.

Added the condition in fq as initial release data is 2006*. Which means all the data whose initial release data is beginning with 2006. Limited the field in the output by adding the necessary attribute values in the fl field.


cloudera-quickstart-vm-5.13.0-0-virtualbox [Running] - Oracle VM VirtualBox
Solr Admin - Mozilla Firefox
quickstart.cloudera:8983/solr/#/films_shard1_replica1/query
/select
q
:
fq
initial_release_date_t:2006*
sort
start, rows
0
10
fl
directed_by_t, initial_release_date_t
df
Raw Query Parameters
key1=val1&key2=val2
wt
json
indent
debugQuery
dismax
edismax
hl
facet
spatial
spellcheck
Execute Query
{
"responseHeader": {
"status": 0,
"QTime": 2,
"params": {
"fl": "directed_by_t, initial_release_date_t",
"indent": "true",
"q": "*:*",
"_": "1678157939703",
"wt": "json",
"fq": "initial_release_date_t:2006*"
}
},
"response": {
"numFound": 173,
"start": 0,
"docs": [
{
"directed_by_t": "Gary Lennon",
"initial_release_date_t": "2006-11-30"
},
{
"directed_by_t": "Zack Snyder",
"initial_release_date_t": "2006-12-09"
},
{
"directed_by_t": "John Lafia",
"initial_release_date_t": "2006-03-18"
},
{
"directed_by_t": "Robert Moresco",
"initial_release_date_t": "2006-08-18"
},
{
"directed_by_t": "Richard Donner",
"initial_release_date_t": "2006-03-01"
},
{
"directed_by_t": "Thom Fitzgerald",
"initial_release_date_t": "2006-12-01"
},