PARALLEL ALGORITHMS

TEAM 3

SNEHA DIDIGAM

SIRISHA MANTRAMURTHY

DEEPTHI PENMETSA

**Q1. Submit Cheat sheet generated through the In-Class Exercises. Identify the Best Indexing Techniques in Parallel Manner and Best Searching Techniques in Sequential and Parallel Manner.**

***BEST INDEXING TECHNIQUES IN PARALLEL MANNER***:

**1.SEPARATE CHAINING OR OPEN HASHING**

In Worst case, O(N)

In Average case, O(NH)

Where N is number of Items

H is the size of hash table

* Uses lists
* A single query takes O(n) pointer dereferences in Separate chaining
* Look up time of O(n/m) {n=elements and m=buckets}
* Insertion never fails
* Extra space is required to store pointers

**2. OPEN ADDRESSING OR CLOSED HASHING**

**LINEAR PROBING**

**Serial Approach Parallel Approach**

**Look up** Best Case: O(n) Best Case: O(1)

Worst Case: O(n2) Worst Case: O(n)

**Insert** Best Case: O(n) Best Case: O(1)

Worst Case: O(n2) Worst Case: O(n)

**Removal** Best Case: O(n) Best Case: O(1)

Worst Case: O(n2) Worst Case: O(n)

* S = (h(k) + i) mod m {key k, hash function h(k), size m}
* All the elements lie in same array
* Insertion is easy

**QUADRATIC PROBING**

* S = (h(k) + i2) mod m {key k, hash function h(k), size m}
* Worst case for searching O(nlogn) or O(logn)
* Access becomes inefficient at higher load factor
* Insertion fails sometimes

**DOUBLE HASHING**

* Two hash functions are used
* Access becomes inefficient at higher load factor

**3.CUCKOO HASHING**

* Hashing with worst-case O(1) lookups
* Deletions take time O(1) because only 2 locations must be checked
* Expected worst case look-up is O(log n)

ADVANCED INDEXING TECHNIQUES

**1.BLOOM FILTER**

* Time Complexity for Insertion and Search is O(k)
* Space Complexity is O(m) {M= bits , k hash function}

**2.INVERTED INDEX**

* Inverted index is an data structure storing a mapping form content
* Inverted file index, list of references to documents for all words
* Full inverted index, positions of each word is also referred in documents

**3.BIG TABLE**

* Maps row key, column key and timestamp
* Tables are split into tablets through GFS
* It has low-latency storage stack

The best indexing technique in parallel manner is Quadratic probing because

* Quadratic probing avoids primary clustering
* Simple storage is required (no dynamic location)
* Faster Insertion when compared to other techniques
* Insertion never fails

***BEST SEARCHING TECHNIQUES IN SEQUENTIAL PARALLEL MANNER***

**1.RANGE SEARCHING**

* Ford-dimensional range trees the Space and Time complexity is O(n(logn)d-1)
* O(logdn+k) is the search complexity in worst case
* Single-Shot:
  + Space S(n) and Time T(n)
* **Re**petitive-Mode:
  + Preprocessing time: P(n)
  + Space occupied by Data Structures: S(n)
  + Query Time: Q(n)
  + Dynamic Case: Update Time U(n)

**2.SEARCH IN MAP REDUCE**

* `uses “map”, “shuffle” and “reduce” functions
* Scalability is achieved
* Flexible and fast
* Simple and implements parallel programming

**Q2. Design a parallel indexing and searching technique and implement it in spark.**

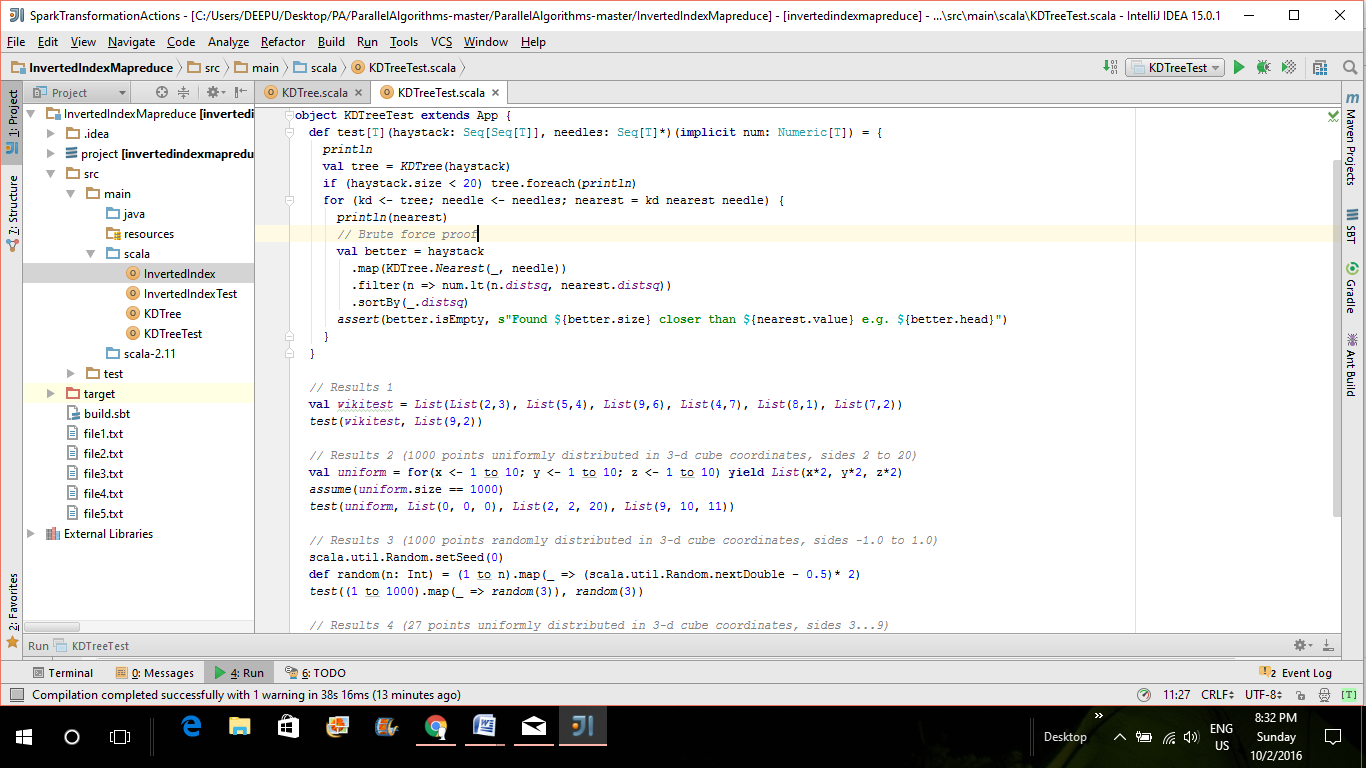
Implemented KD Trees

KD Tree is a data structure for organizing points in K dimensional space. They are useful for several applications like searches involving in a multidimensional search key (Range queries and nearest neighbor search). They are special case of binary space portioning keys.

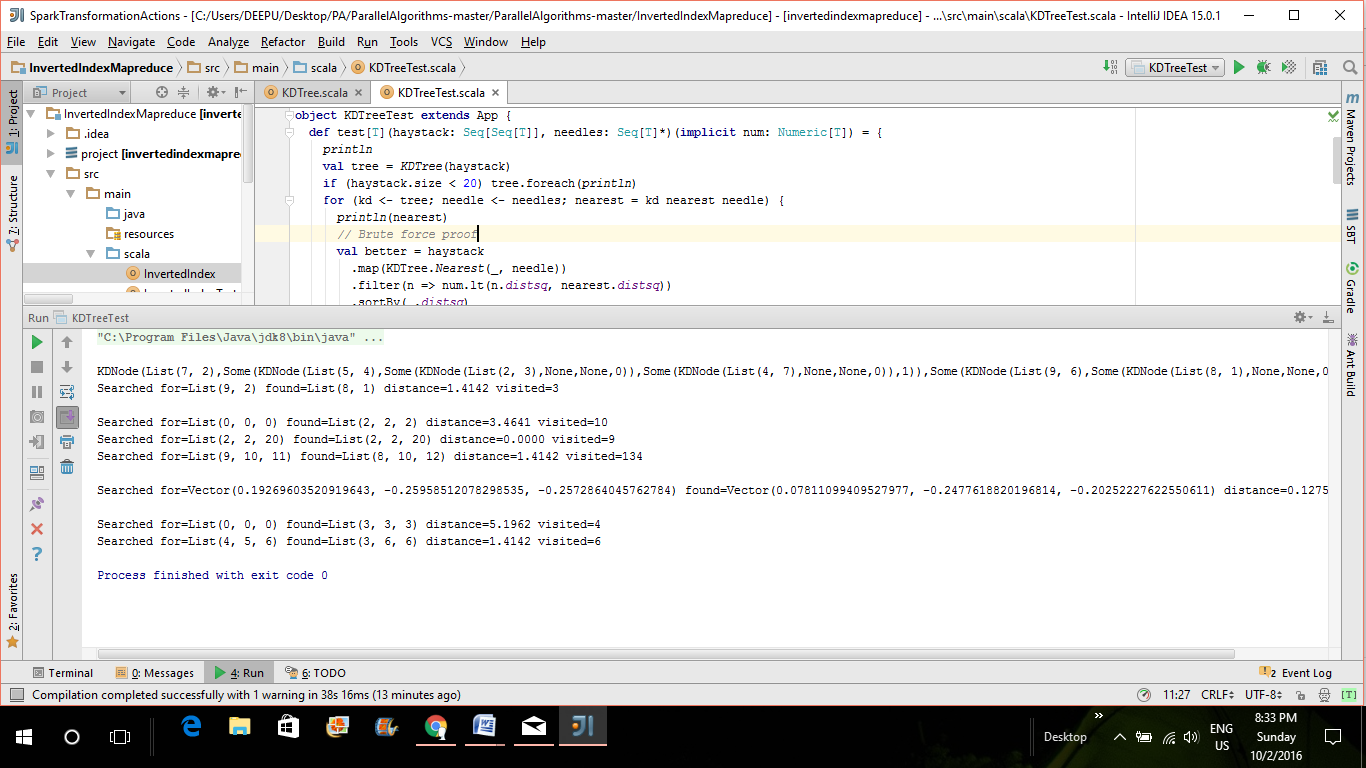
So we constructed a KD tree and perform nearest neighbor search for the sample dataset as shown below

The Wikipedia example data of [(2,3), (5,4), (9,6), (4,7), (8,1), (7,2)].

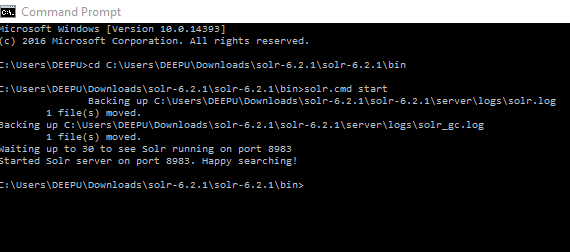
For the Wikipedia example above , we will find the nearest neighbor to point (9, 2) For the random data, pick a random location and will find the nearest neighbor.

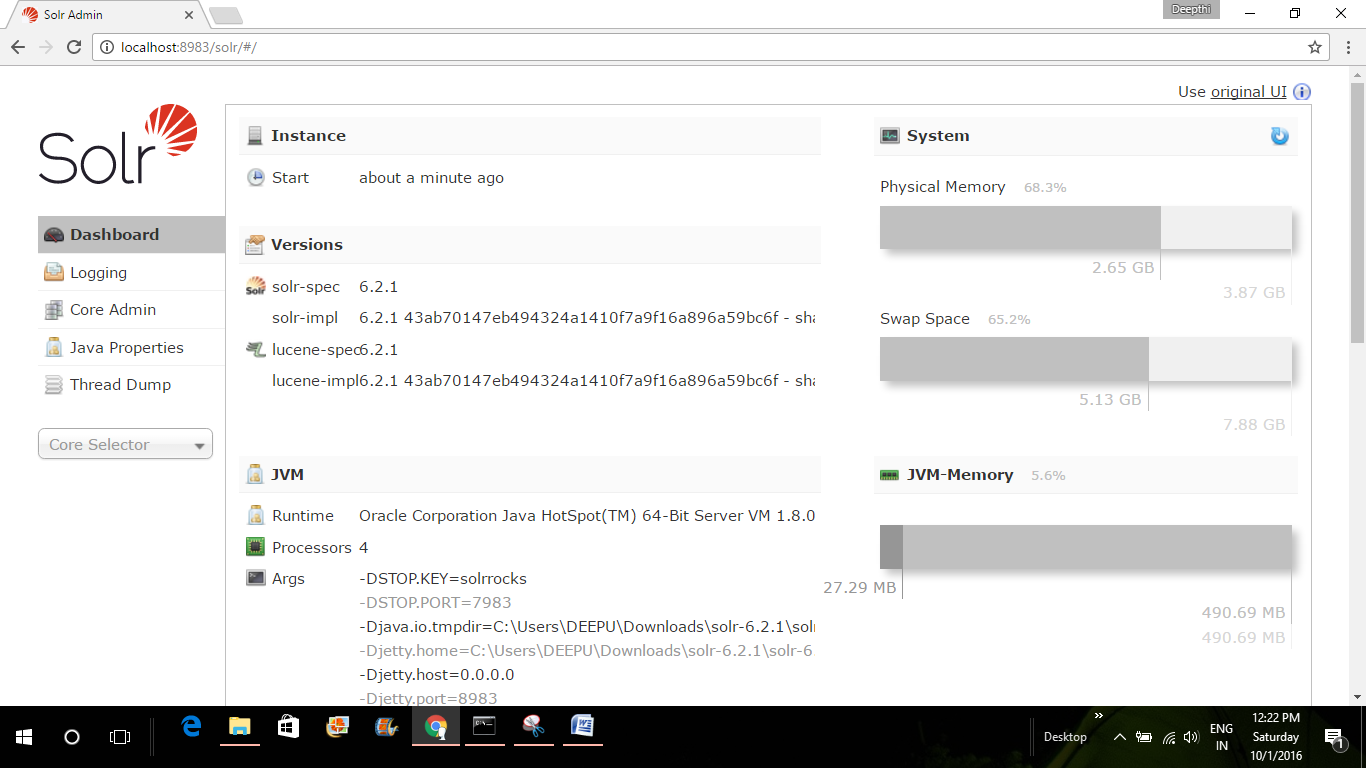


Output



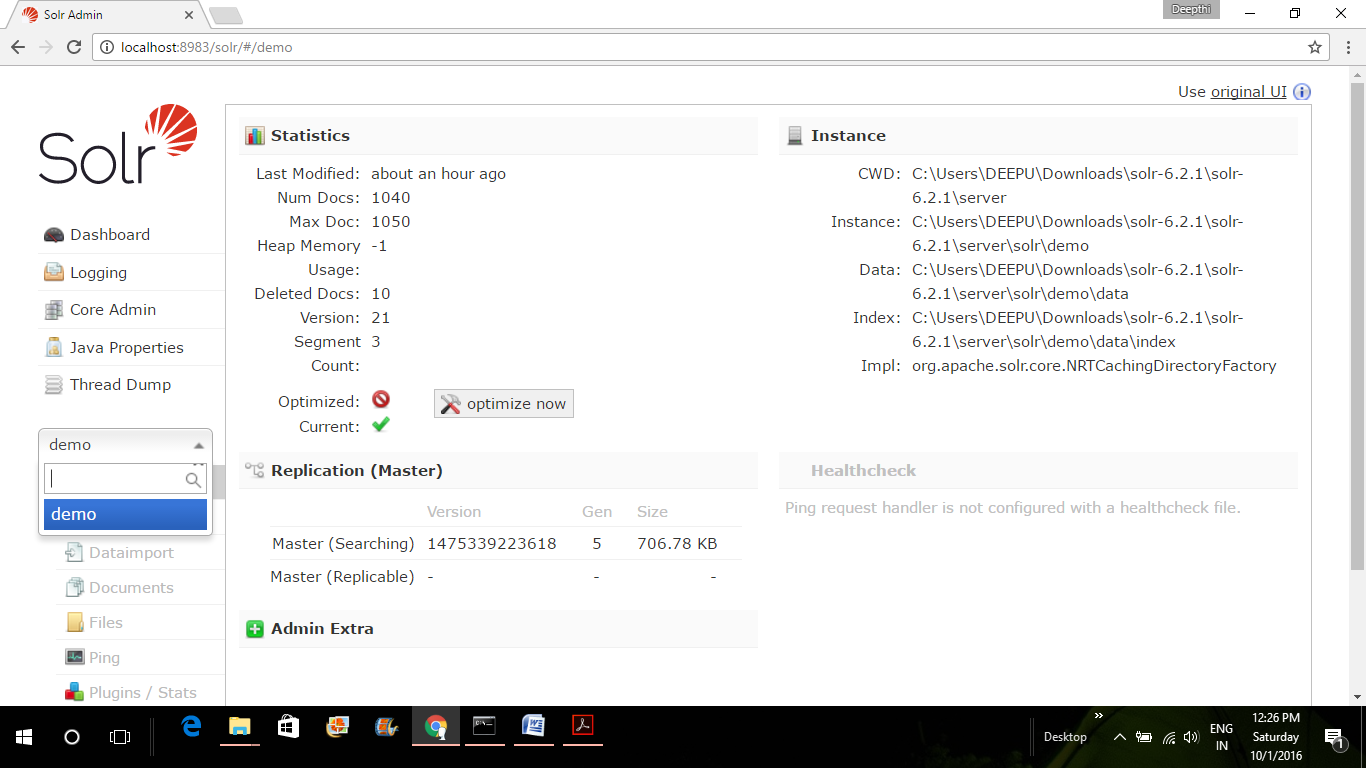
**Q3. Apache Solr:**





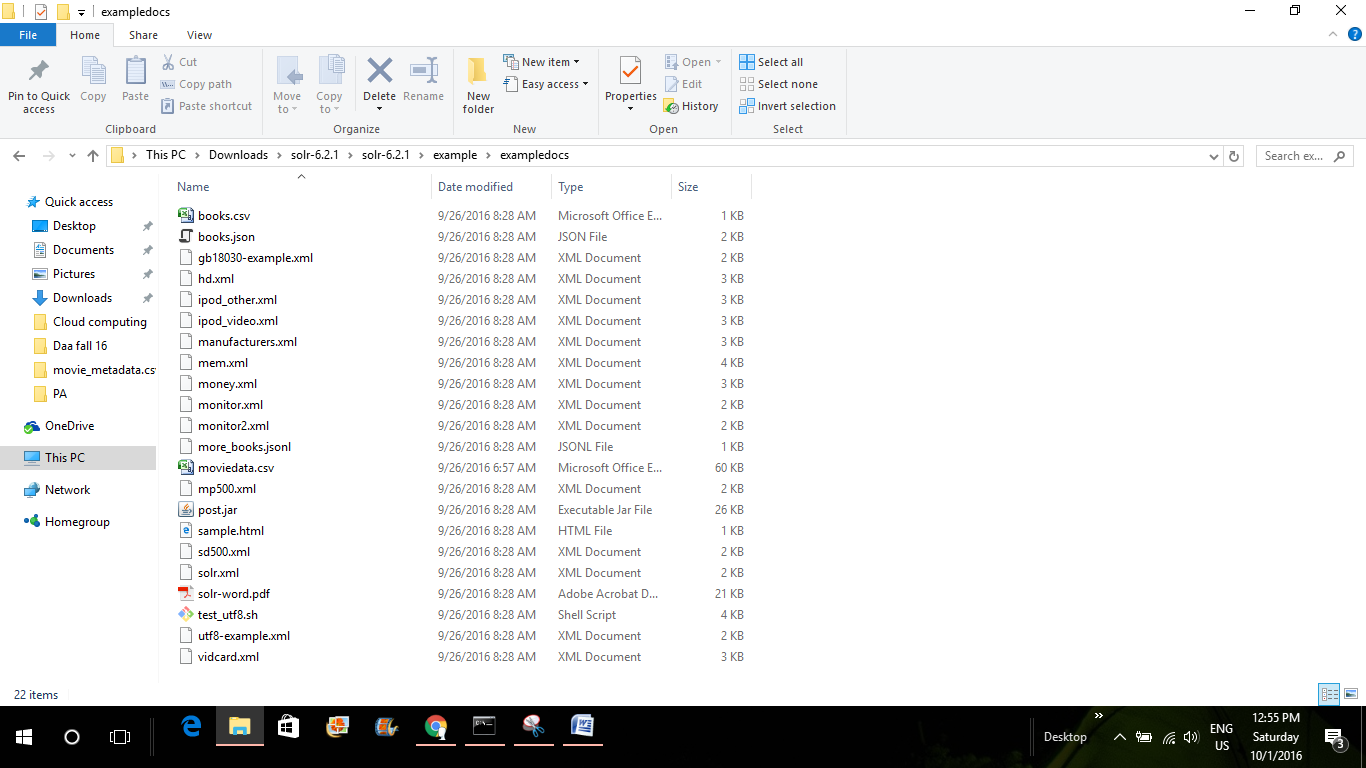
Then created a core in order to be able to do index and search with the following command in command prompt

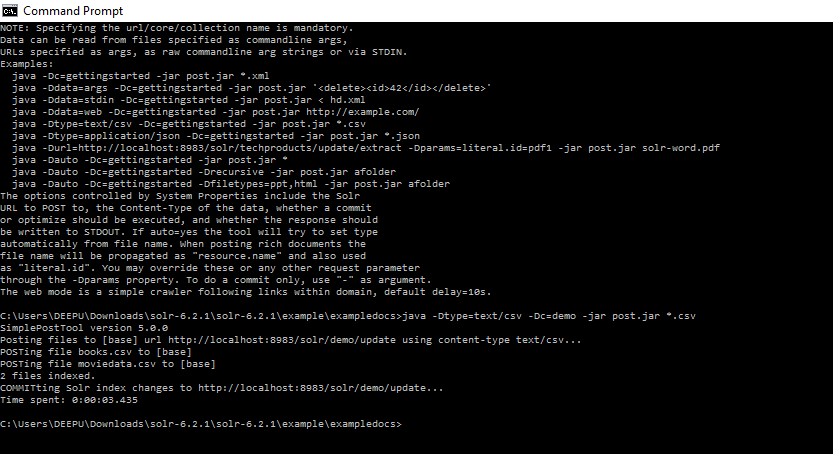
$ bin/solr create -c demo



Now index files in any format(csv/json/pdf/text/xml/folder)

I have indexed a csv file named moviedata.csv





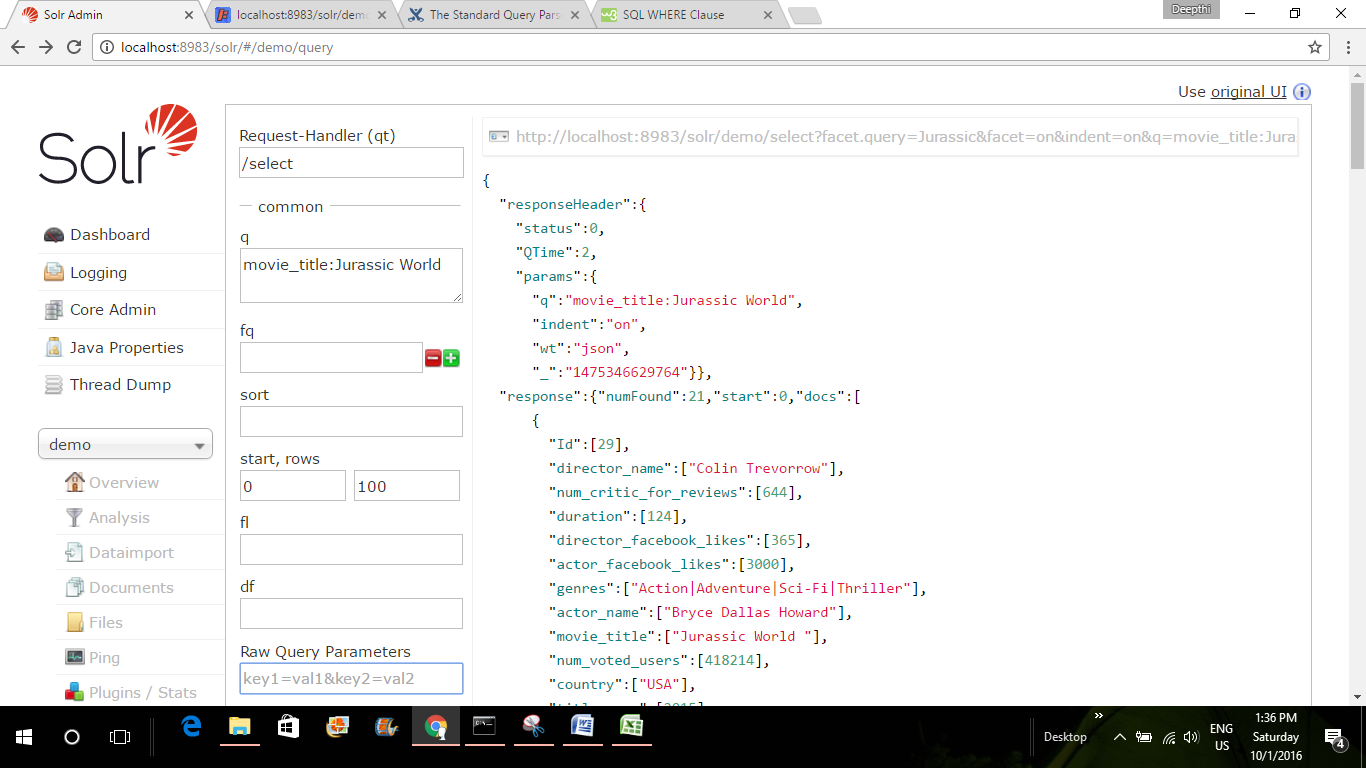
Queries and their responses

1. List of rows with movie\_title Jurassic world.

The response also includes with rows having word as either Jurassic or World or even both.

<http://localhost:8983/solr/demo/select?indent=on&q=movie_title:Jurassic%20World&rows=100&start=0&wt=csv>

It lists all the rows which has the word either Jurassic or world, or even both.

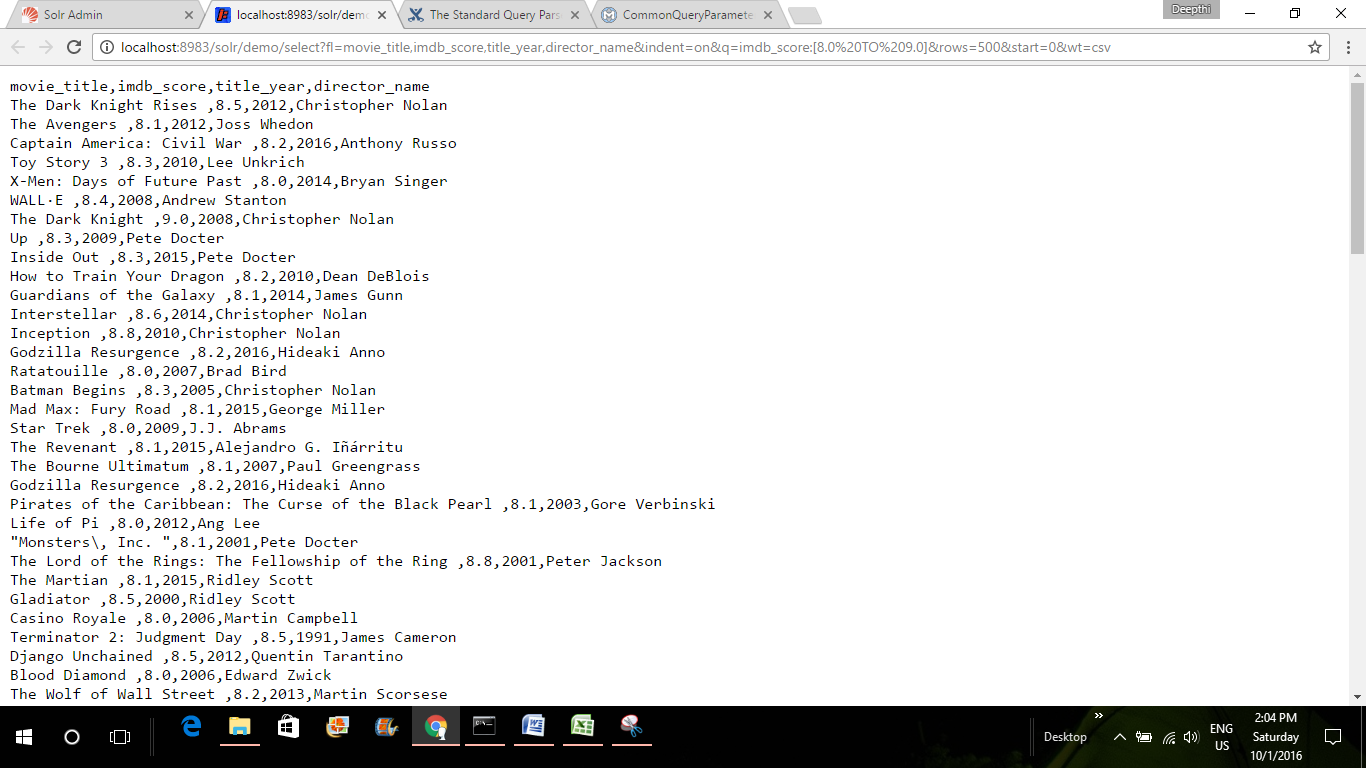




2. List of movies with imdb\_score in the range 8.0 and 9.0 along with year of release and director of the movie

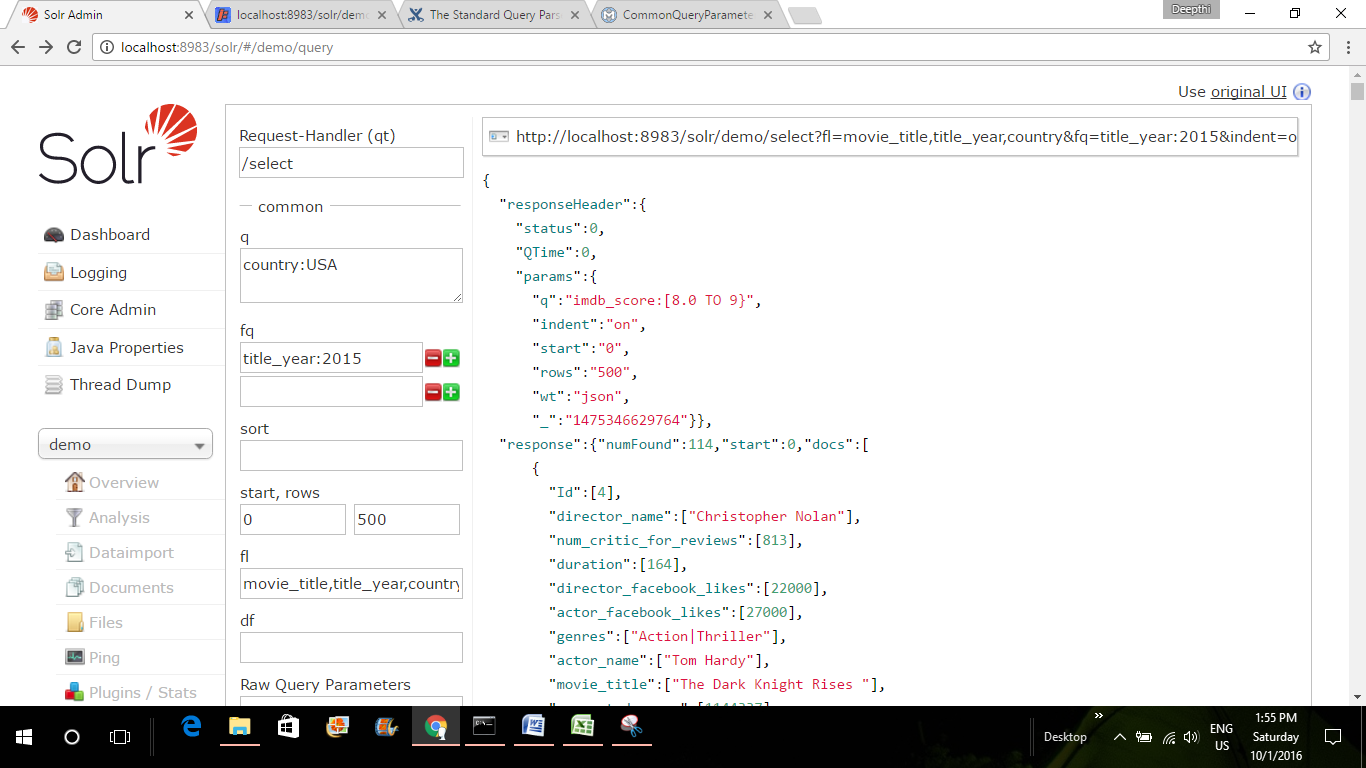
[http://localhost:8983/solr/demo/select?fl=movie\_title,imdb\_score,title\_year,director\_name&indent=on&q=imdb\_score:[8.0%20TO%209.0]&rows=500&start=0&wt=csv](http://localhost:8983/solr/demo/select?fl=movie_title,imdb_score,title_year,director_name&indent=on&q=imdb_score:%5b8.0%20TO%209.0%5d&rows=500&start=0&wt=csv)

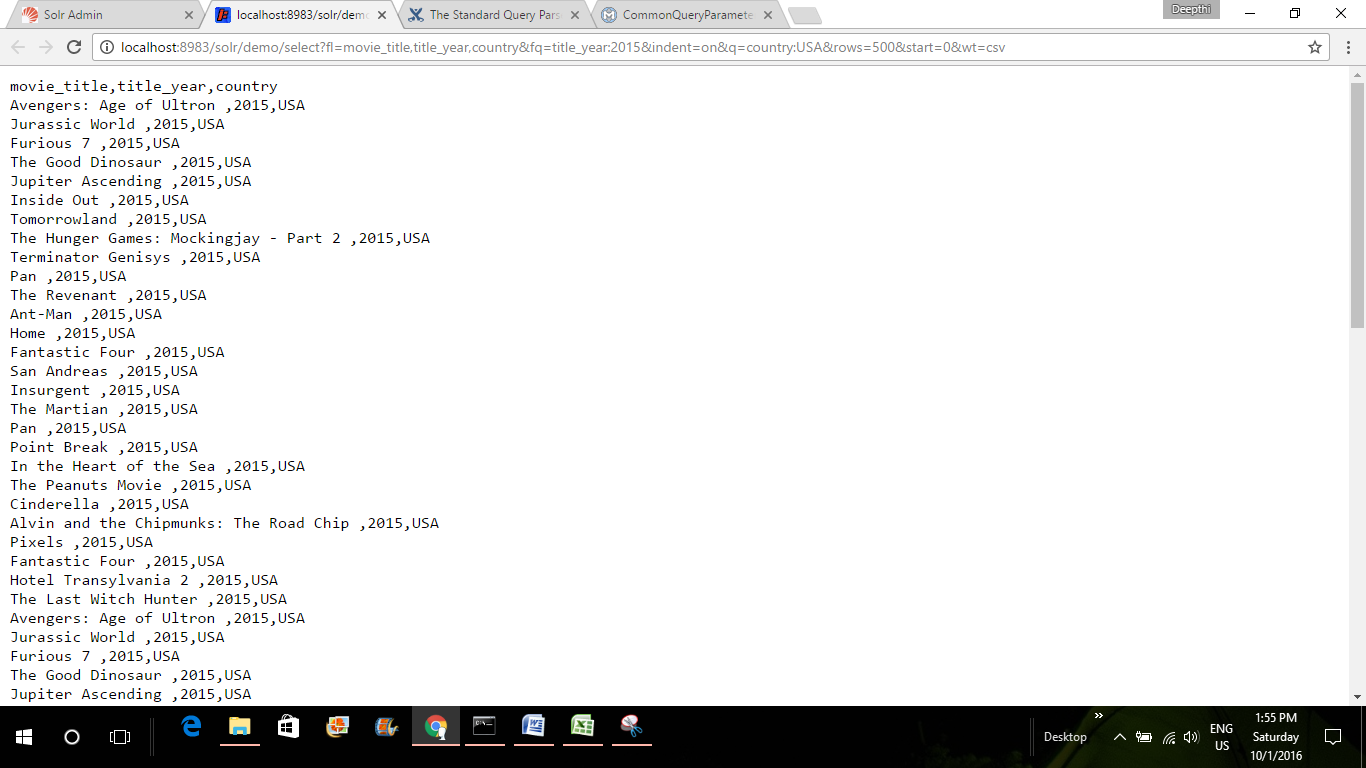




3. List of movies released in year 2015 in country USA

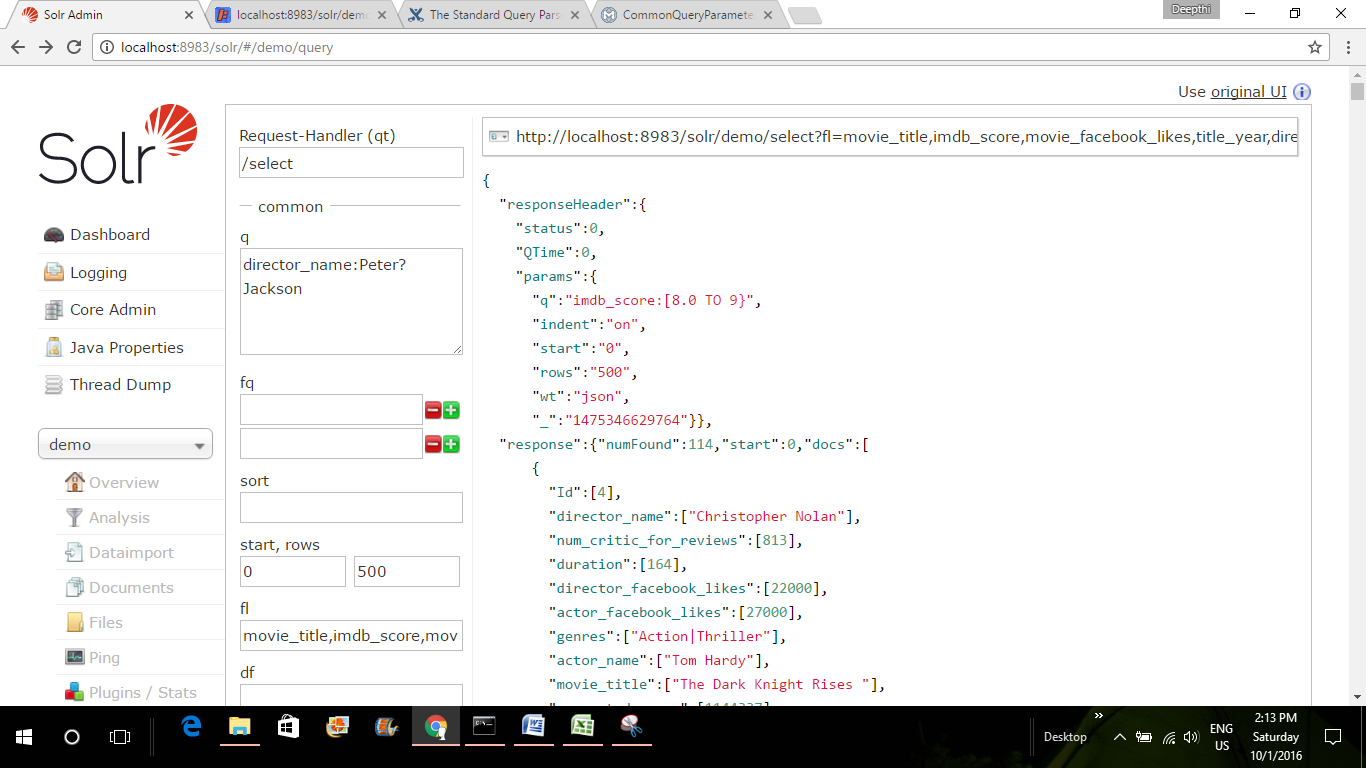
<http://localhost:8983/solr/demo/select?fl=movie_title,title_year,country&fq=title_year:2015&indent=on&q=country:USA&rows=500&start=0&wt=csv>

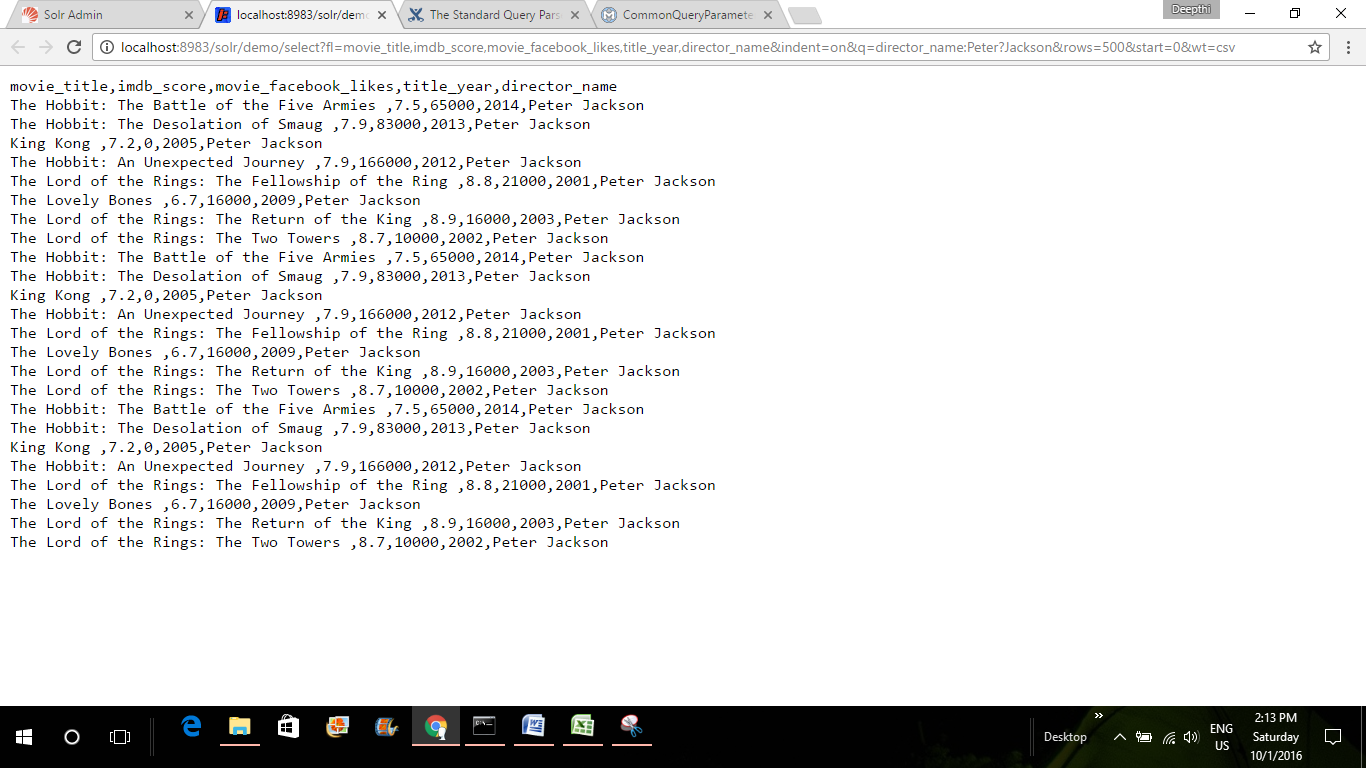




4. List of movies directed by peter Jackson along with imdb\_score, year of release and movie facebook likes

<http://localhost:8983/solr/demo/select?fl=movie_title,imdb_score,movie_facebook_likes,title_year,director_name&indent=on&q=director_name:Peter?Jackson&rows=500&start=0&wt=csv>





5. List of director\_name,movie\_title,imdb\_score and country of the movies with genre type: Action|Adventure|Fantasy|Sci-Fi|Thriller

<http://localhost:8983/solr/demo/select?fl=director_name,movie_title,title_year,imdb_score,country&indent=on&q=genres:Action|Adventure|Fantasy|Sci-Fi|Thriller&rows=500&start=0&wt=csv>

