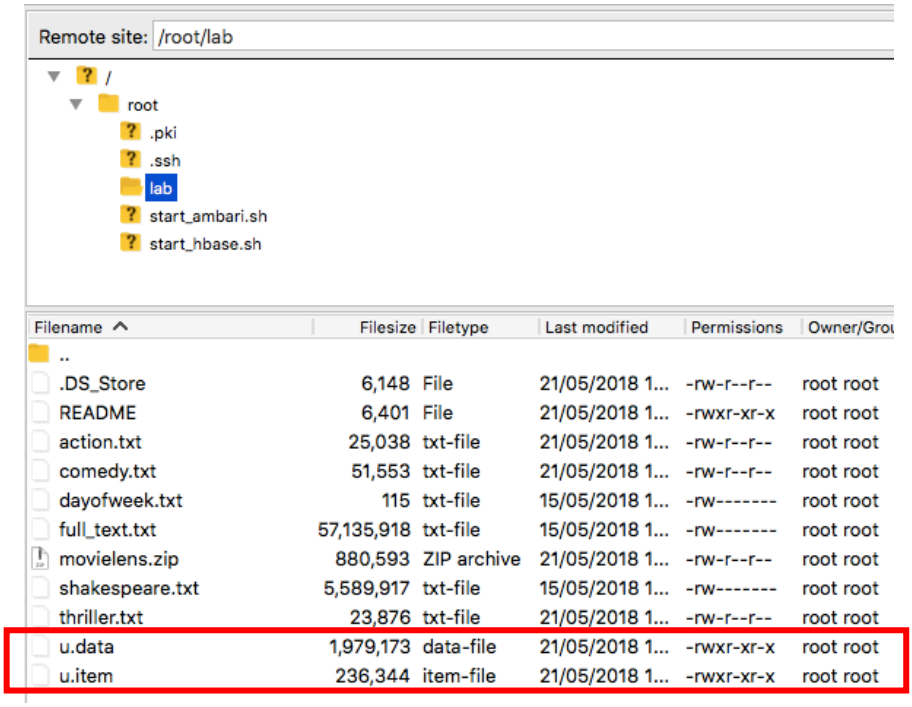
CIND 719 - ASSIGNMENT 1 - PROGRAMMING HIVE
NAME: DIANA MOYANO

We first need to move u.data and u.item files from the local site to the virtual machine (in this case, Linux)



Files have been moved to the virtual machine through FileZilla and the following command is written to move it into HDFS

$ hadoop fs -put /root/lab/u.item /user/lab/u.item

```
[root@sandbox lab]# hadoop fs -cat /user/lab/u.item | head -n 5
1|Toy Story (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Toy%20Story%20(1995)|0|0|0|1|1|1|0|0|0|0|0|0|0|0|0|0|0|0|0
2|GoldenEye (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?GoldenEye%20(1995)|0|1|1|0|0|0|0|0|0|0|0|0|0|0|0|1|0|0
3|Four Rooms (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Four%20Rooms%20(1995)|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|1|0|0
4|Get Shorty (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Get%20Shorty%20(1995)|0|1|0|0|0|1|0|0|1|0|0|0|0|0|0|0|0|0
5|Copycat (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Copycat%20(1995)|0|0|0|0|0|0|1|0|1|0|0|0|0|0|0|1|0|0
cat: Unable to write to output stream.
[root@sandbox lab]#
```

$ hadoop fs -put /root/lab/u.data /user/lab/u.data

```
[root@sandbox lab]# hadoop fs -put /root/lab/u.data /user/lab/u.data
[root@sandbox lab]# hadoop fs -cat /user/lab/u.data | head -n 5
196     242     3       881250949
186     302     3       891717742
22      377     1       878887116
244     51      2       880606923
166     346     1       886397596
```

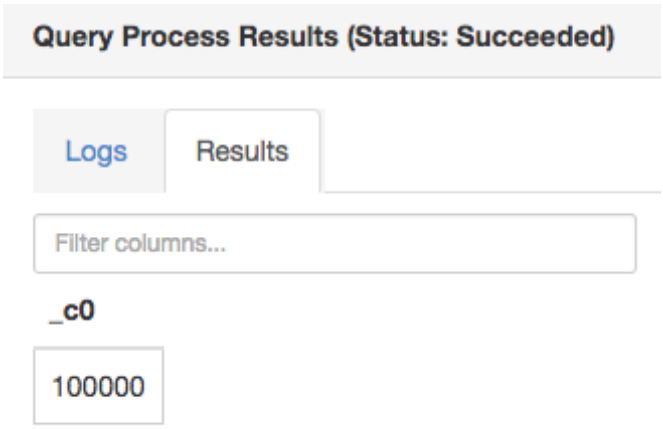
In order to create both tables (ratings and movies), the following command is entered:

CREATE DATABASE ml;

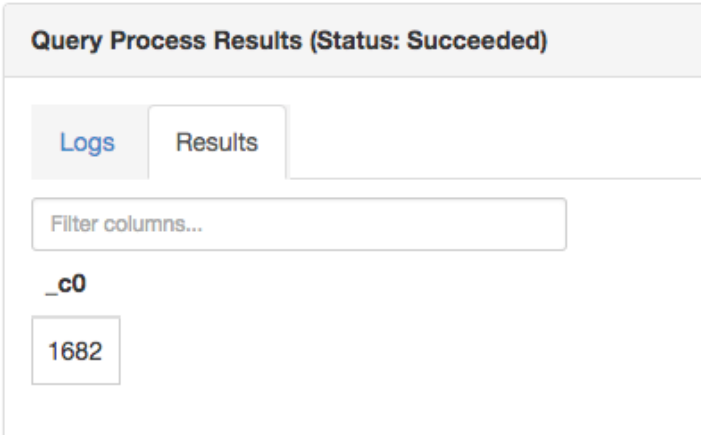
Movies table	Ratings Table
CREATE TABLE ml.movies **(movieid INT,** **movie_title STRING,** **release_date STRING,** **v_release_date STRING,** **imdb_url STRING,** **cat_unknown INT,** **cat_action INT,** **cat_adventure INT,** **cat_animation INT,** **cat_children INT,** **cat_comedy INT,** **cat_crime INT,** **cat_documentary INT,** **cat_drama INT,** **cat_fantasy INT,** **cat_fill_noir INT,** **cat_horror INT,** **cat_musical INT,** **cat_mystery INT,** **cat_romance INT,** **cat_scifi INT,** **cat_thriller INT,** **cat_war INT,** **cat_western INT)** **ROW FORMAT DELIMITED** **FIELDS TERMINATED BY '|'** **STORED AS TEXTFILE;** **LOAD DATA INPATH '/user/lab/u.item'** **INTO TABLE ml.movies;**	**CREATE TABLE ml.userratings** **(userid INT, movieid INT, rating INT,** **unixtime BIGINT) ROW FORMAT DELIMITED** **FIELDS TERMINATED BY '\t'** **STORED AS TEXTFILE;** **LOAD DATA INPATH '/user/lab/u.data'** **INTO TABLE ml.userratings;**

1. How many records are there in both tables? Please specify separately for each table.

select count(*) from ml.userratings;



select count(*) from ml.movies;



2. Find the name of all movies released in 1990.

```
select movie_title from ml.movies
where substr(release_date,8,4)=1990;
```

movie_title
Home Alone (1990)
Dances with Wolves (1990)
GoodFellas (1990)
Nikita (La Femme Nikita) (1990)
Cyrano de Bergerac (1990)
Die Hard 2 (1990)
Hunt for Red October, The (1990)
Ghost (1990)
Amityville Curse, The (1990)
Miller's Crossing (1990)
Grifters, The (1990)
Paris Is Burning (1990)
Rosencrantz and Guildenstern Are Dead (1990)
Pump Up the Volume (1990)
Pretty Woman (1990)
Days of Thunder (1990)
Tie Me Up! Tie Me Down! (1990)
Trust (1990)
Young Guns II (1990)
Marked for Death (1990)
Every Other Weekend (1990)
I, Worst of All (Yo, la peor de todas) (1990)
American Dream (1990)
King of New York (1990)

3. List the movieid of the 10 films that received the most ratings (not necessarily highest rating) in the table you created from u.data.

```
select movieid, count(userid) as count_rev
from ml.userratings
group by movieid
order by count_rev desc
limit 10;
```

movieid	count_rev
50	583
258	509
100	508
181	507
294	485
286	481
288	478
1	452
300	431
121	429

4. Use a join to list the titles of the movies you found in step 3.

```
create table ml.movies_join1 as
select b.movieid, a.movie_title, b.userid
from ml.movies a JOIN ml.userratings b
ON b.movieid = a.movieid;
```

movies_join1.movieid	movies_join1.movie_title	movies_join1.userid
242	Kolya (1996)	196
302	L.A. Confidential (1997)	186
377	Heavyweights (1994)	22
51	Legends of the Fall (1994)	244
346	Jackie Brown (1997)	166

```
select movie_title, movieid, count(userid) as
count_rev
from ml.movies_join1
group by movie_title, movieid
order by count_rev desc
limit 10;
```

movie_title	movieid	count_rev
Star Wars (1977)	50	583
Contact (1997)	258	509
Fargo (1996)	100	508
Return of the Jedi (1983)	181	507
Liar Liar (1997)	294	485
English Patient, The (1996)	286	481
Scream (1996)	288	478
Toy Story (1995)	1	452
Air Force One (1997)	300	431
Independence Day (ID4) (1996)	121	429

5. Find the highest rated sci_fi movie. Explain how you define "highest rating".

```
create table ml.movies_join2 as
select a.movie_title, a.cat_scifi, b.rating
from ml.movies a JOIN ml.userratings b
ON b.movieid = a.movieid;
```

movies_join2.movie_title	movies_join2.cat_scifi	movies_join2.rating
Kolya (1996)	0	3
L.A. Confidential (1997)	0	3
Heavyweights (1994)	0	1
Legends of the Fall (1994)	0	2
Jackie Brown (1997)	0	1
Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1963)	1	4

```
select movie_title, avg(rating) as average_rating, count(rating) as count_reviews
from ml.movies_join2
where cat_scifi =1
group by movie_title
order by average_rating desc
limit 10;
```

movie_title	average_rating	count_ratings
Star Kid (1997)	5.0	3
Star Wars (1977)	4.3584905660377355	583
Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1963)	4.252577319587629	194
Empire Strikes Back, The (1980)	4.204359673024523	367
Blade Runner (1982)	4.138181818181818	275
Alien (1979)	4.034364261168385	291
Return of the Jedi (1983)	4.007889546351085	507
Terminator 2: Judgment Day (1991)	4.0067796610169495	295
2001: A Space Odyssey (1968)	3.969111969111969	259
Aliens (1986)	3.9471830985915495	284

An average rating is obtained per movie title and finally sorted in descending order. However, the result above suggests that the number of ratings per movie should be also considered, as we cannot simply consider Star Kid as the movie with highest rating, given that movies with way higher number of ratings such as Star Wars should be more relevant for the purpose of this query. In order to deal with this, we can add one more condition related to a minimum number of ratings per movie.

For now, I'd chose Star Wars (1977)

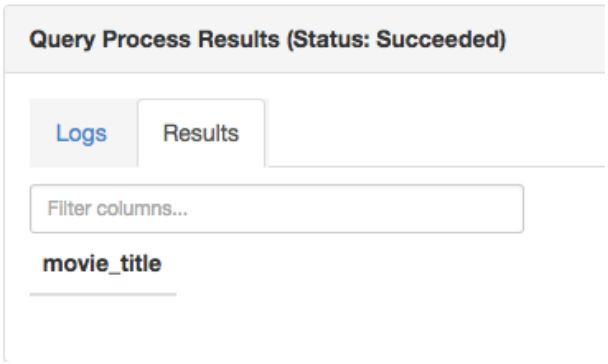
BONUS: Are there any movies with no ratings? (Hint: outer join and IS NULL)

We first create a table that includes the outer left join, being the movies table on the left, so it will show all the movie titles with its ratings.

```
create table ml.movies_join5 as
select a.movie_title, b.rating
from ml.movies a LEFT OUTER JOIN ml.userratings b
ON a.movieid = b.movieid;
```

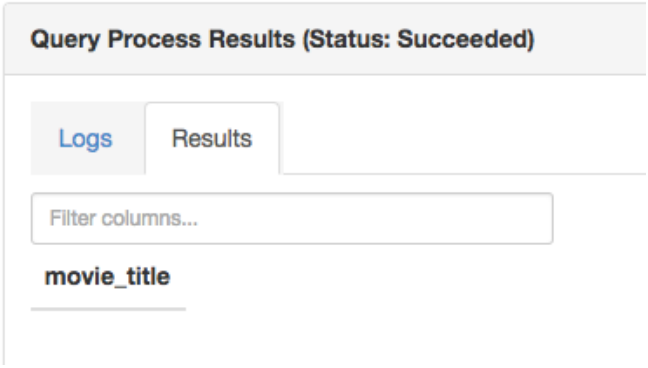
Once done, the following query will look for the movie titles that have null values

```
select movie_title
from ml.movies_join5
where rating is NULL;
```



Another way to do this that do not require to create a table is by writing the following query:

```
SELECT movie_title
   FROM ml.movies a
   LEFT OUTER JOIN ml.userratings
b ON (a.movieid=b.movieid)
   WHERE b.movieid IS NULL;
```



Both methods suggest that there are no movies with no ratings