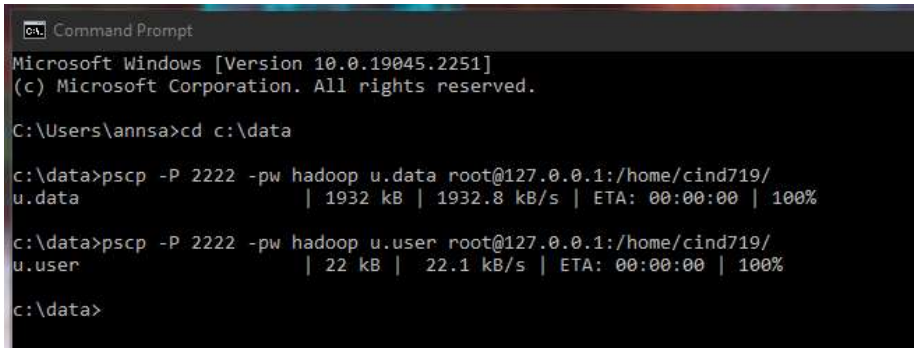
Question 1:

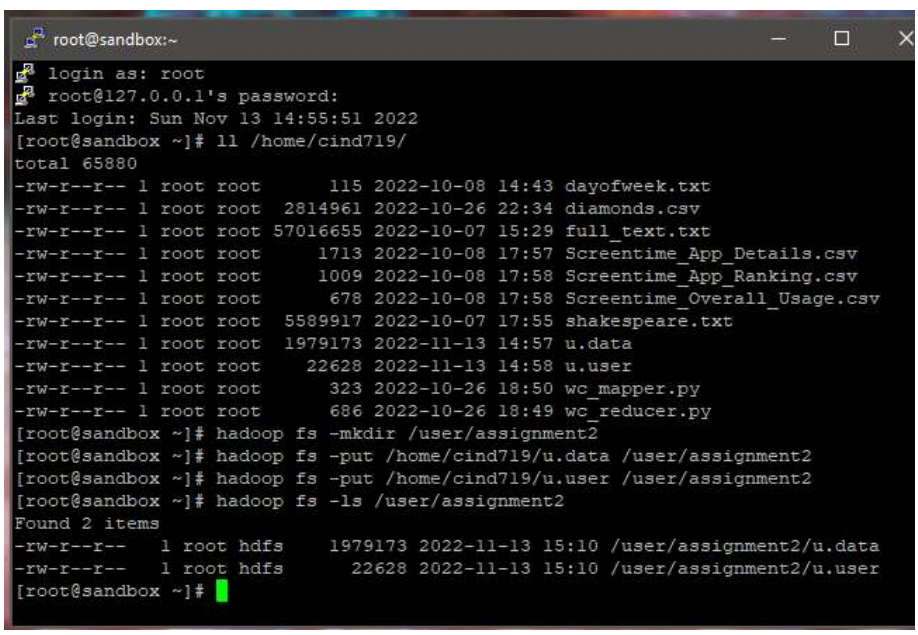
Store the data in a Hive database ml as table userratings(u.data), users (u.user)

Import instructions

- Transferring the files from host OS to HDP access node using the pspc utility in Windows Command Prompt:
```
pscp -P 2222 -pw hadoop u.data root@127.0.0.1:/home/cind719/
pscp -P 2222 -pw hadoop u.user root@127.0.0.1:/home/cind719/
```
-

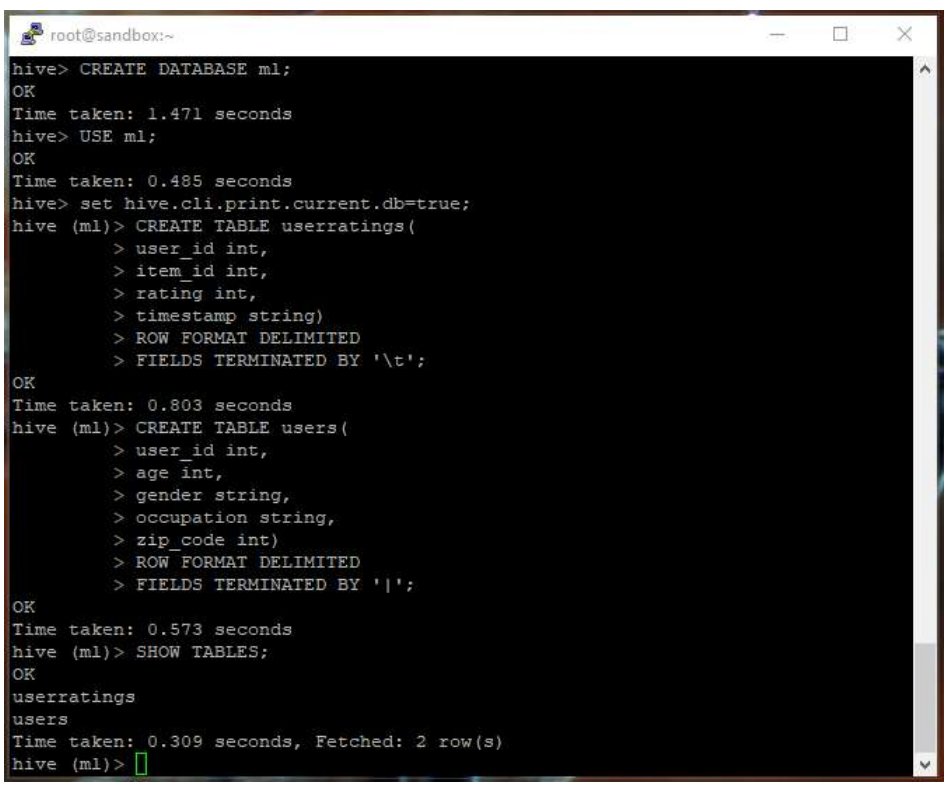


- Creating a new directory using the HDP terminal and moving the files to the HDFS:
```
hadoop fs -mkdir /user/assignment2
hadoop fs -put /home/cind719/u.data /user/assignment2
hadoop fs -put /home/cind719/u.user /user/assignment2
```



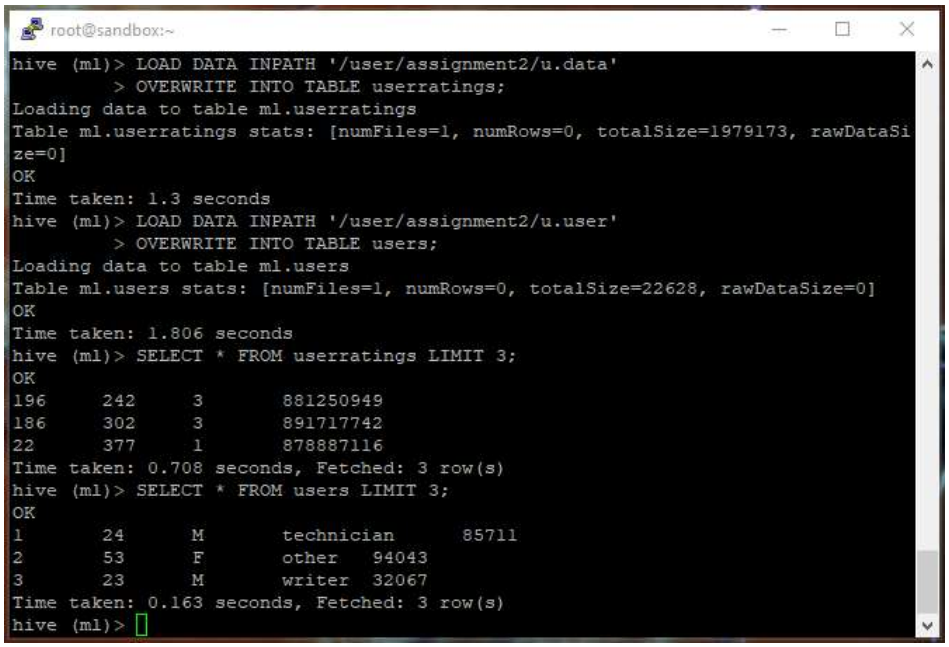
- Database and table creation scripts:

```
CREATE DATABASE ml;
USE ml;
set hive.cli.print.current.db=true;

CREATE TABLE userratings(
      user_id int,
      item_id int,
      rating int,
      timestamp string)
      ROW FORMAT DELIMITED
      FIELDS TERMINATED BY '\t';

CREATE TABLE users(
      user_id int,
      age int,
      gender string,
      occupation string,
      zip_code int)
      ROW FORMAT DELIMITED
      FIELDS TERMINATED BY '|';
```



- Importing csv files into tables:

```
LOAD DATA INPATH '/user/assignment2/u.data'
OVERWRITE INTO TABLE userratings;

LOAD DATA INPATH '/user/assignment2/u.user'
OVERWRITE INTO TABLE users;
```



```
root@sandbox:~
hive (ml)> LOAD DATA INPATH '/user/assignment2/u.data'
         > OVERWRITE INTO TABLE userratings;
Loading data to table ml.userratings
Table ml.userratings stats: [numFiles=1, numRows=0, totalSize=1979173, rawDataSi
ze=0]
OK
Time taken: 1.3 seconds
hive (ml)> LOAD DATA INPATH '/user/assignment2/u.user'
         > OVERWRITE INTO TABLE users;
Loading data to table ml.users
Table ml.users stats: [numFiles=1, numRows=0, totalSize=22628, rawDataSize=0]
OK
Time taken: 1.806 seconds
hive (ml)> SELECT * FROM userratings LIMIT 3;
OK
196     242     3       881250949
186     302     3       891717742
22      377     1       878887116
Time taken: 0.708 seconds, Fetched: 3 row(s)
hive (ml)> SELECT * FROM users LIMIT 3;
OK
1       24      M       technician      85711
2       53      F       other   94043
3       23      M       writer  32067
Time taken: 0.163 seconds, Fetched: 3 row(s)
hive (ml)>
```

<u>Question 2:</u>

Write HiveQL queries to confirm the number of records in both tables.

- Number of records in userratings table is 100000

```
SELECT COUNT(*) FROM userratings;
```

```
root@sandbox:~
hive (ml)> SELECT COUNT(*) FROM userratings;
Query ID = root_20221113162525_dc1976e6-b4c0-4a74-b78c-2e02baa413b8
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.


Status: Running (Executing on YARN cluster with App id application_1668351282007
_0010)

--------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      1          1        0        0       0       0
Reducer 2 ......   SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 8.94 s
--------------------------------------------------------------------------------
OK
100000
Time taken: 22.11 seconds, Fetched: 1 row(s)
hive (ml)>
```

- Number of records in users table is 943

```
SELECT COUNT(*) FROM users;
```

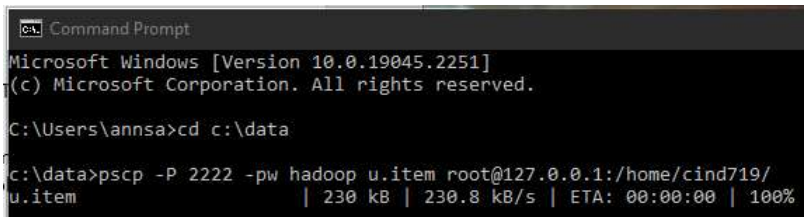
```
root@sandbox:~
hive (ml)> SELECT COUNT(*) FROM users;
Query ID = root_20221113162929_3c5a4fa1-fc85-4f61-aefa-d15aeaf4923d
Total jobs = 1
Launching Job 1 out of 1


Status: Running (Executing on YARN cluster with App id application_1668351282007
_0010)

--------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      1          1        0        0       0       0
Reducer 2 ......   SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 8.00 s
--------------------------------------------------------------------------------
OK
943
Time taken: 9.387 seconds, Fetched: 1 row(s)
hive (ml)>
```

Question 3:

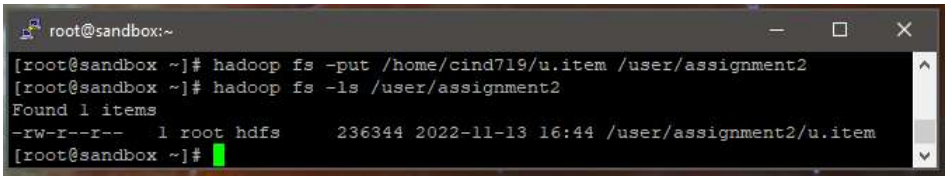
Extract the list of top 10 items (movies) that received the most ratings from male educators.

- Importing u.item file and creating table movies

```
pscp -P 2222 -pw hadoop u.item root@127.0.0.1:/home/cind719/
```




```
hadoop fs -put /home/cind719/u.item /user/assignment2
```




```
CREATE TABLE movies(
      movie_id int,
      movie_title string,
      release_date string,
      video_release_date string,
      imdb_url string,
      unknown int,
      action int,
      adventure int,
      animation int,
      childrens int,
      comedy int,
      crime int,
      documentary int,
      drama int,
      fantasy int,
      film_noir int,
      horror int,
      musical int,
      mystery int,
      romance int,
      scifi int,
      thriller int,
      war int,
      western int)
      ROW FORMAT DELIMITED
      FIELDS TERMINATED BY '|';
```

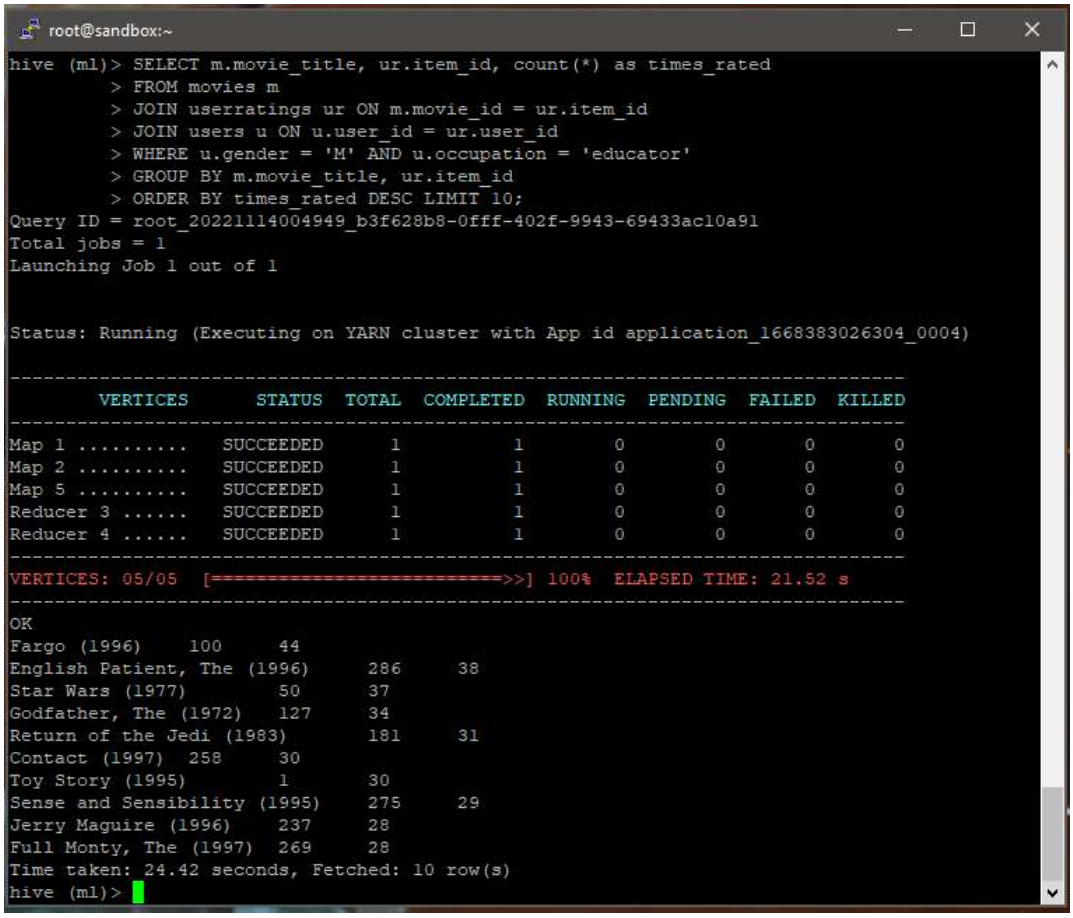
```
root@sandbox:~
hive (ml)> CREATE TABLE movies(
         > movie_id int,
         > movie_title string,
         > release_date string,
         > video_release_date string,
         > imdb_url string,
         > unknown int,
         > action int,
         > adventure int,
         > animation int,
         > childrens int,
         > comedy int,
         > crime int,
         > documentary int,
         > drama int,
         > fantasy int,
         > film_noir int,
         > horro int,
         > musical int,
         > mystery int,
         > romance int,
         > scifi int,
         > thriller int,
         > war int,
         > western int)
         > ROW FORMAT DELIMITED
         > FIELDS TERMINATED BY '|';
OK
Time taken: 0.739 seconds
```

```
LOAD DATA INPATH '/user/assignment2/u.item'
OVERWRITE INTO TABLE movies;
```

```
root@sandbox:~
hive (ml)> LOAD DATA INPATH '/user/assignment2/u.item'
         > OVERWRITE INTO TABLE movies;
Loading data to table ml.movies
Table ml.movies stats: [numFiles=1, numRows=0, totalSize=236344, rawDataSize=0]
OK
Time taken: 1.3 seconds
hive (ml)> SELECT * FROM movies LIMIT 3;
OK
1       Toy Story (1995)        01-Jan-1995             http://us.imdb.com/M/tit
le-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/tit
le-exact?GoldenEye%20(1995)     0       1       1       0       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/tit
le-exact?Four%20Rooms%20(1995)  0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0       1       0
0
Time taken: 0.926 seconds, Fetched: 3 row(s)
hive (ml)>
```

- Query top 10 items (movies) with the most ratings from male educators

```
SELECT m.movie_title, ur.item_id, count(*) as times_rated
FROM movies m
JOIN userratings ur ON m.movie_id = ur.item_id
JOIN users u ON u.user_id = ur.user_id
WHERE u.gender = 'M' AND u.occupation = 'educator'
GROUP BY m.movie_title, ur.item_id
ORDER BY ratings DESC LIMIT 10;
```



```
root@sandbox:~
hive (ml)> SELECT m.movie_title, ur.item_id, count(*) as times_rated
         > FROM movies m
         > JOIN userratings ur ON m.movie_id = ur.item_id
         > JOIN users u ON u.user_id = ur.user_id
         > WHERE u.gender = 'M' AND u.occupation = 'educator'
         > GROUP BY m.movie_title, ur.item_id
         > ORDER BY times_rated DESC LIMIT 10;
Query ID = root_20221114004949_b3f628b8-0fff-402f-9943-69433ac10a91
Total jobs = 1
Launching Job 1 out of 1


Status: Running (Executing on YARN cluster with App id application_1668383026304_0004)

--------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      1          1        0        0       0       0
Map 2 ..........   SUCCEEDED      1          1        0        0       0       0
Map 5 ..........   SUCCEEDED      1          1        0        0       0       0
Reducer 3 ......   SUCCEEDED      1          1        0        0       0       0
Reducer 4 ......   SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 05/05  [==========================>>] 100%  ELAPSED TIME: 21.52 s
--------------------------------------------------------------------------------
OK
Fargo (1996)    100     44
English Patient, The (1996)     286     38
Star Wars (1977)        50      37
Godfather, The (1972)   127     34
Return of the Jedi (1983)       181     31
Contact (1997)  258     30
Toy Story (1995)        1       30
Sense and Sensibility (1995)    275     29
Jerry Maguire (1996)    237     28
Full Monty, The (1997)  269     28
Time taken: 24.42 seconds, Fetched: 10 row(s)
hive (ml)>
```

Question 4:

Find the highest rated Fantasy movie

- Star Kid (1997) with an average rating of 5

```
SELECT m.movie_title, AVG(ur.rating) as avg_rating
FROM movies m
JOIN userratings ur ON m.movie_id = ur.item_id
WHERE m.fantasy = '1'
GROUP BY m.movie_title
ORDER BY avg_rating DESC LIMIT 1;
```

```
root@sandbox:~
hive (ml)> SELECT m.movie_title, AVG(ur.rating) as avg_rating
         > FROM movies m
         > JOIN userratings ur ON m.movie_id = ur.item_id
         > WHERE m.fantasy = '1'
         > GROUP BY m.movie_title
         > ORDER BY avg_rating DESC LIMIT 1;
Query ID = root_20221118035353_8f8d1cc6-e5c7-4894-86c7-fe5d3d69d0a6
Total jobs = 1
Launching Job 1 out of 1


Status: Running (Executing on YARN cluster with App id application_1668742616965
_0001)

--------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      1          1        0        0       0       0
Map 4 ..........   SUCCEEDED      1          1        0        0       0       0
Reducer 2 ......   SUCCEEDED      1          1        0        0       0       0
Reducer 3 ......   SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 04/04  [==========================>>] 100%  ELAPSED TIME: 14.52 s
--------------------------------------------------------------------------------
OK
Star Kid (1997) 5.0
Time taken: 16.007 seconds, Fetched: 1 row(s)
hive (ml)>
```

Question 5

Load the u.data and u.user files into Apache Spark as DataFrames named df_udata and df_uuser:

- Command Prompt:
  ```
  pscp -P 2020 -pw spark u.data cu@127.0.0.1:/home/cu/data/
  pscp -P 2020 -pw spark u.user cu@127.0.0.1:/home/cu/data/
  ```

```
Command Prompt
Microsoft Windows [Version 10.0.19045.2251]
(c) Microsoft Corporation. All rights reserved.

C:\Users\annsa>cd c:\data

c:\data>pscp -P 2020 -pw spark u.data cu@127.0.0.1:/home/cu/data/
u.data                    | 1932 kB | 1932.8 kB/s | ETA: 00:00:00 | 100%

c:\data>pscp -P 2020 -pw spark u.user cu@127.0.0.1:/home/cu/data/
u.user                    | 22 kB |   22.1 kB/s | ETA: 00:00:00 | 100%

c:\data>
```

- Apache Spark:
  ```
  df_udata =
  spark.read.format("csv").option("delimiter","\t").option("quote","").sc
  hema("user_id integer, item_id integer, rating integer, timestamp
  integer").load("/home/cu/data/u.data")
  ```

```
cu@spark3cuvm: ~
SparkSession available as 'spark'.
>>> df_udata = spark.read.format("csv").option("delimiter","\t").option("quote","").
schema("user_id integer, item_id integer, rating integer, timestamp integer").load("
/home/cu/data/u.data")
>>> df_udata.show(10)
+-------+-------+------+---------+
|user_id|item_id|rating|timestamp|
+-------+-------+------+---------+
|    196|    242|     3|881250949|
|    186|    302|     3|891717742|
|     22|    377|     1|878887116|
|    244|     51|     2|880606923|
|    166|    346|     1|886397596|
|    298|    474|     4|884182806|
|    115|    265|     2|881171488|
|    253|    465|     5|891628467|
|    305|    451|     3|886324817|
|      6|     86|     3|883603013|
+-------+-------+------+---------+
only showing top 10 rows

>>>
```

```
df_uuser =
spark.read.format("csv").option("delimiter","|").option("quote","").sch
ema("user_id integer, age integer, gender string, occupation string,
zip_code integer").load("/home/cu/data/u.user")
```

```
cu@spark3cuvm: ~
>>> df_uuser = spark.read.format("csv").option("delimiter","|").option("quote","").s
chema("user_id integer, age integer, gender string, occupation string, zip_code inte
ger").load("/home/cu/data/u.user")
>>> df_uuser.show(10)
+-------+---+------+-------------+--------+
|user_id|age|gender|   occupation|zip_code|
+-------+---+------+-------------+--------+
|      1| 24|     M|   technician|   85711|
|      2| 53|     F|        other|   94043|
|      3| 23|     M|       writer|   32067|
|      4| 24|     M|   technician|   43537|
|      5| 33|     F|        other|   15213|
|      6| 42|     M|    executive|   98101|
|      7| 57|     M|administrator|   91344|
|      8| 36|     M|administrator|    5201|
|      9| 29|     M|      student|    1002|
|     10| 53|     M|       lawyer|   90703|
+-------+---+------+-------------+--------+
only showing top 10 rows

>>>
```

- **How many unique occupations are in the data and what is the frequency of each occupation?**

  First register the data frames with the Spark engine as a table:

```
>>> df_udata.registerTempTable('ml_udata_table')
>>> df_uuser.registerTempTable('ml_uuser_table')
```

```
cu@spark3cuvm: ~
>>> df_udata.registerTempTable('ml_udata_table')
>>> df_uuser.registerTempTable('ml_uuser_table')
>>>
```

Running the SQL query:

```
>>> df_unique_occupations = spark.sql("SELECT occupation, COUNT(*) as
frequency FROM ml_uuser_table GROUP BY occupation ORDER BY frequency
DESC")
>>> df_unique_occupations.show()
```

```
cu@spark3cuvm: ~
>>> df_unique_occupations = spark.sql("SELECT occupation, COUNT(*) as frequency
FROM ml_uuser_table GROUP BY occupation")
>>> df_unique_occupations.show()
+-------------+---------+
|   occupation|frequency|
+-------------+---------+
|    librarian|       51|
|      retired|       14|
|       lawyer|       12|
|         none|        9|
|       writer|       45|
|   programmer|       66|
|    marketing|       26|
|        other|      105|
|    executive|       32|
|    scientist|       31|
|      student|      196|
|     salesman|       12|
|       artist|       28|
|   technician|       27|
|administrator|       79|
|     engineer|       67|
|   healthcare|       16|
|     educator|       95|
|entertainment|       18|
|    homemaker|        7|
+-------------+---------+
only showing top 20 rows

>>>
```

- **Find the number of recommendations corresponding to each occupation.**

```
>>> rec_by_occupation = spark.sql("SELECT u.occupation,
COUNT(ur.rating) as recommendations FROM ml_uuser_table u JOIN
ml_udata_table ur ON u.user_id == ur.user_id GROUP BY u.occupation")
>>> rec_by_occupation.show()
```

```
cu@spark3cuvm: ~
>>> rec_by_occupation = spark.sql("SELECT u.occupation, count(ur.rating) as recommen
N ml_udata_table ur ON u.user_id == ur.user_id GROUP BY u.occupation")
>>> rec_by_occupation.show()
+-------------+----------------+
|   occupation|recommendations|
+-------------+----------------+
|    librarian|           5273|
|      retired|           1609|
|       lawyer|           1345|
|         none|            901|
|       writer|           5536|
|   programmer|           7801|
|    marketing|           1950|
|        other|          10663|
|    executive|           3403|
|    scientist|           2058|
|      student|          21957|
|     salesman|            856|
|       artist|           2308|
|   technician|           3506|
|administrator|           7479|
|     engineer|           8175|
|   healthcare|           2804|
|     educator|           9442|
|entertainment|           2095|
|    homemaker|            299|
+-------------+----------------+
only showing top 20 rows

>>>
```