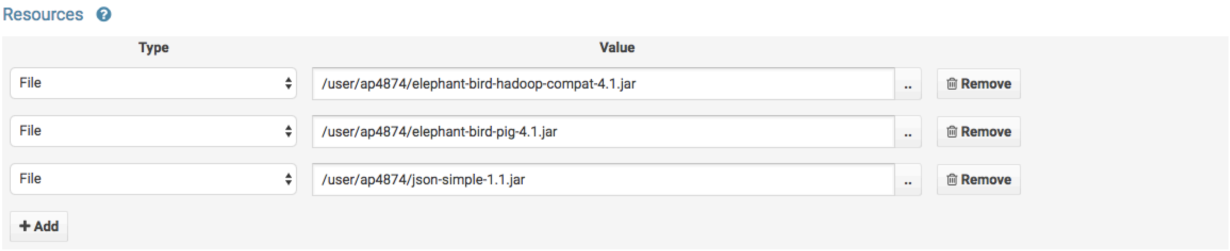
BIG DATA PROGRAMMING

ASSIGNMENT 2

Submitted By: Abhinish Popli
Email: ap4874@nyu.edu

To load the json files into the Pig, we should use Twitter's open source Library Elephant Bird. To do so, we need to add jar files in the Pig Properties:

a) elephant-bird-hadoop-compat-4.1.jar
b) elephant-bird-pig-4.1.jar
c) json-simple-1.1.jar

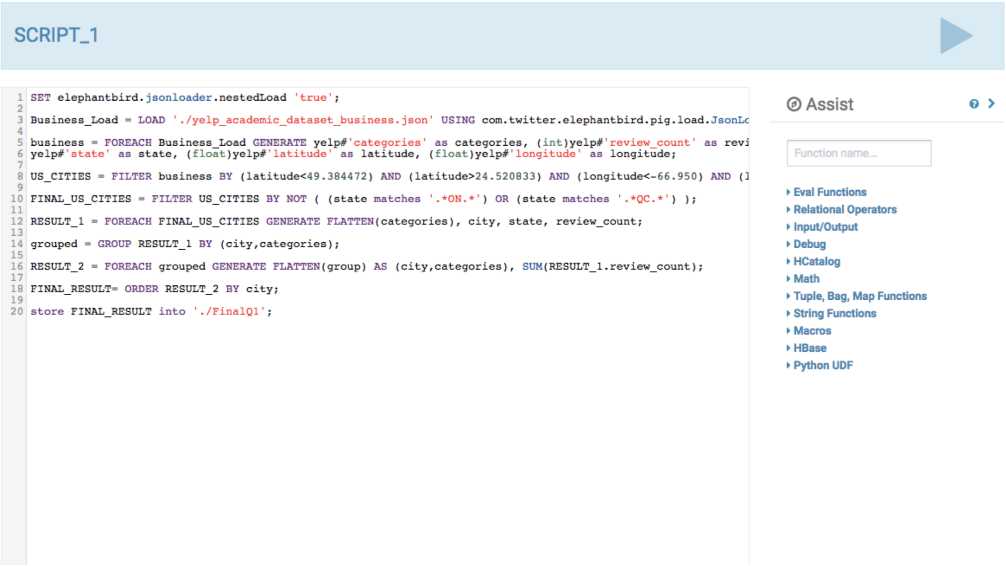


Resources

Type	Value
File	/user/ap4874/elephant-bird-hadoop-compat-4.1.jar
File	/user/ap4874/elephant-bird-pig-4.1.jar
File	/user/ap4874/json-simple-1.1.jar

Figure 1. Add jar files under the Property Tab.

1. Summarize the number of reviews by US city, by business category.



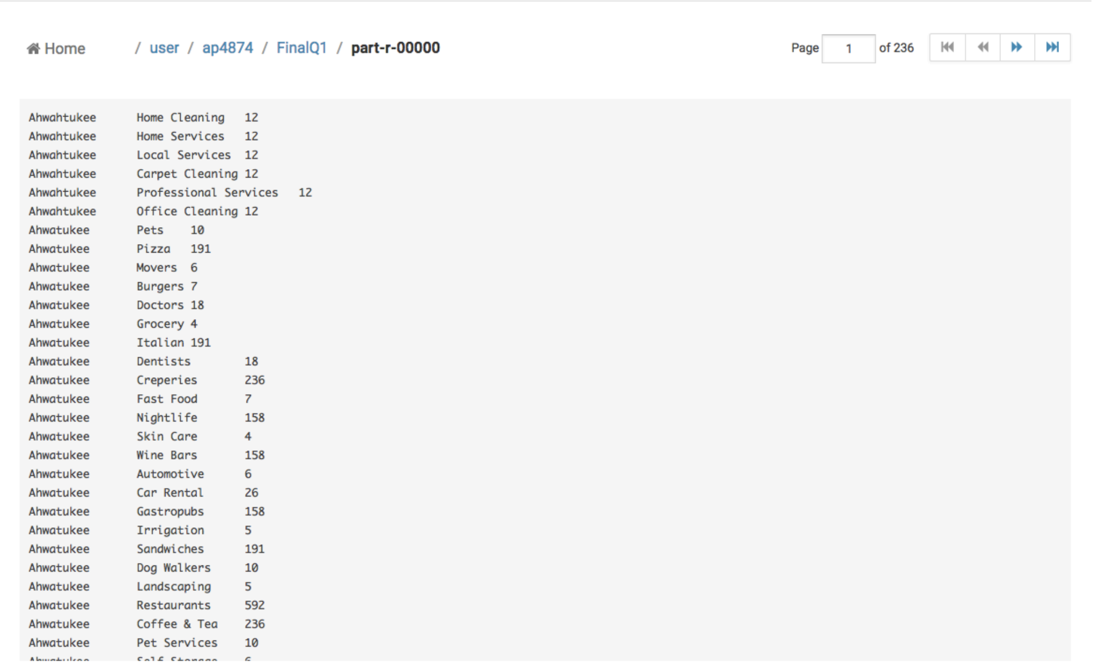
SCRIPT_1

```
SET elephantbird.jsonloader.nestedLoad 'true';

Business_Load = LOAD './yelp_academic_dataset_business.json' USING com.twitter.elephantbird.pig.load.JsonLo

business = FOREACH Business_Load GENERATE yelp#'categories' as categories, (int)yelp#'review_count' as revi
yelp#'state' as state, (float)yelp#'latitude' as latitude, (float)yelp#'longitude' as longitude;

US_CITIES = FILTER business BY (latitude<49.384472) AND (latitude>24.520833) AND (longitude<-66.950) AND (l

FINAL_US_CITIES = FILTER US_CITIES BY NOT ( (state matches '.*ON.*') OR (state matches '.*QC.*') );

RESULT_1 = FOREACH FINAL_US_CITIES GENERATE FLATTEN(categories), city, state, review_count;

grouped = GROUP RESULT_1 BY (city,categories);

RESULT_2 = FOREACH grouped GENERATE FLATTEN(group) AS (city,categories), SUM(RESULT_1.review_count);

FINAL_RESULT= ORDER RESULT_2 BY city;

store FINAL_RESULT into './FinalQ1';
```

Figure 2. Pig Script

Explanation:

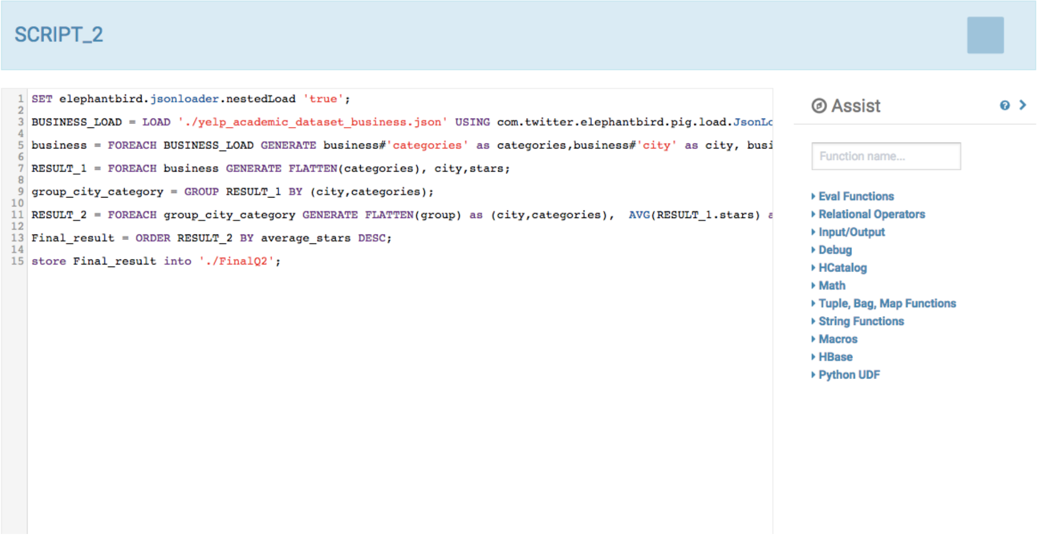
In this Script, first of all json file named “yelp_academic_dataset_business.json” is loaded in Pig with Load command using Elephant Bird Library and stored in the variable “Business_Load”. To fetch specific columns out of the business dataset, I used Foreach command that generate columns ‘categories’, ‘city’, ‘state’, ‘latitude’, ‘longitude’ and ‘review_count’. Then stored it in “business”. To filter out the US Cities, use the ‘filter’ command on latitude and longitude columns i.e. Those cities lie between (latitude<49.384472 and latitude>24.520833) and (longitude<-66.950 and longitude>-124.766667), are fall under US territory. To get more precise results, check if still some cities are not located in US are there in dataset. In our case, there are some that lie in filtered dataset based on latitude and longitude, example: Toronto. To filter those cities, we filter state in the same way we did earlier. As data under attribute ‘categories’ is complex, so to get correct result we need to flatten the data under the given attribute using ‘Flatten ()’ command. To get the final result, I used group by clause on ‘city’ and ‘categories’ and perform sum () operation on ‘review_count’. Then I sorted the data using order by clause on ‘city’. To store the output file, there is a keyword ‘Store’ that I used under directory ‘FinalQ1’.



```
Ahwahtukee      Home Cleaning   12
Ahwahtukee      Home Services   12
Ahwahtukee      Local Services  12
Ahwahtukee      Carpet Cleaning 12
Ahwahtukee      Professional Services   12
Ahwahtukee      Office Cleaning 12
Ahwatukee       Pets    10
Ahwatukee       Pizza   191
Ahwatukee       Movers  6
Ahwatukee       Burgers 7
Ahwatukee       Doctors 18
Ahwatukee       Grocery 4
Ahwatukee       Italian 191
Ahwatukee       Dentists        18
Ahwatukee       Creperies       236
Ahwatukee       Fast Food       7
Ahwatukee       Nightlife       158
Ahwatukee       Skin Care       4
Ahwatukee       Wine Bars       158
Ahwatukee       Automotive      6
Ahwatukee       Car Rental      26
Ahwatukee       Gastropubs      158
Ahwatukee       Irrigation      5
Ahwatukee       Sandwiches      191
Ahwatukee       Dog Walkers     10
Ahwatukee       Landscaping     5
Ahwatukee       Restaurants     592
Ahwatukee       Coffee & Tea    236
Ahwatukee       Pet Services    10
Ahwatukee       Self Storage    6
```

Figure 3. Data in Output File ‘FinalQ1’

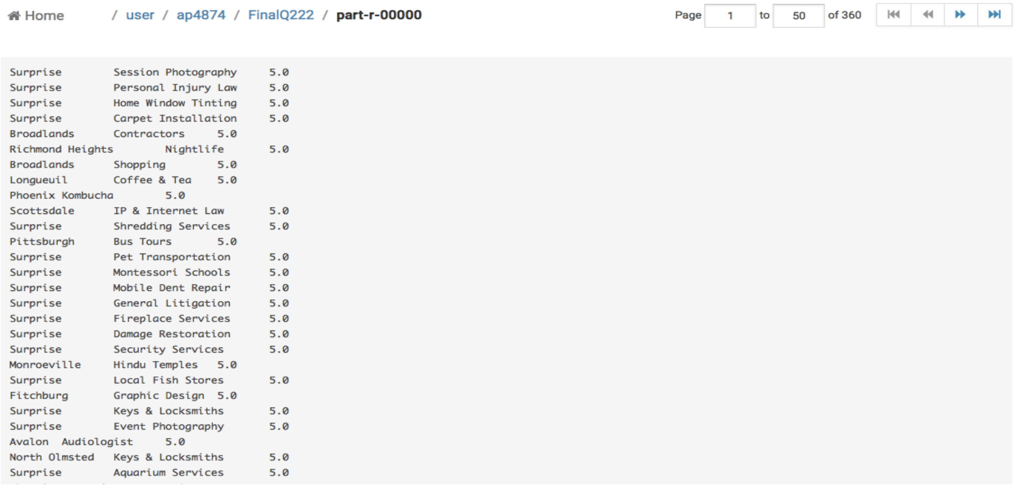
2. Rank all ***cities*** by # of stars descending, for **each category**



```
SET elephantbird.jsonloader.nestedLoad 'true';

BUSINESS_LOAD = LOAD './yelp_academic_dataset_business.json' USING com.twitter.elephantbird.pig.load.JsonLc

business = FOREACH BUSINESS_LOAD GENERATE business#'categories' as categories,business#'city' as city, busi

RESULT_1 = FOREACH business GENERATE FLATTEN(categories), city,stars;

group_city_category = GROUP RESULT_1 BY (city,categories);

RESULT_2 = FOREACH group_city_category GENERATE FLATTEN(group) as (city,categories),  AVG(RESULT_1.stars) a

Final_result = ORDER RESULT_2 BY average_stars DESC;

store Final_result into './FinalQ2';
```

Figure 4. Pig Script

Explanation: To rank all cities according to number of stars for each category, we need data from one file i.e. Business. Load the 'business.json' file in the Pig using Elephant Bird library. To fetch certain columns out of the dataset, I used 'foreach' and 'generate' keywords. After fetching the attributes, I did flattening on the field 'categories' because of its complex data type. Then use the 'group by' clause on city and category and perform 'AVG ()' on stars i.e. AVG(stars) and sorted the data based on average_stars in DESC order, because we need the rank the cities in descending order as given in the query. Then output in stored under directory 'FinalQ2'.



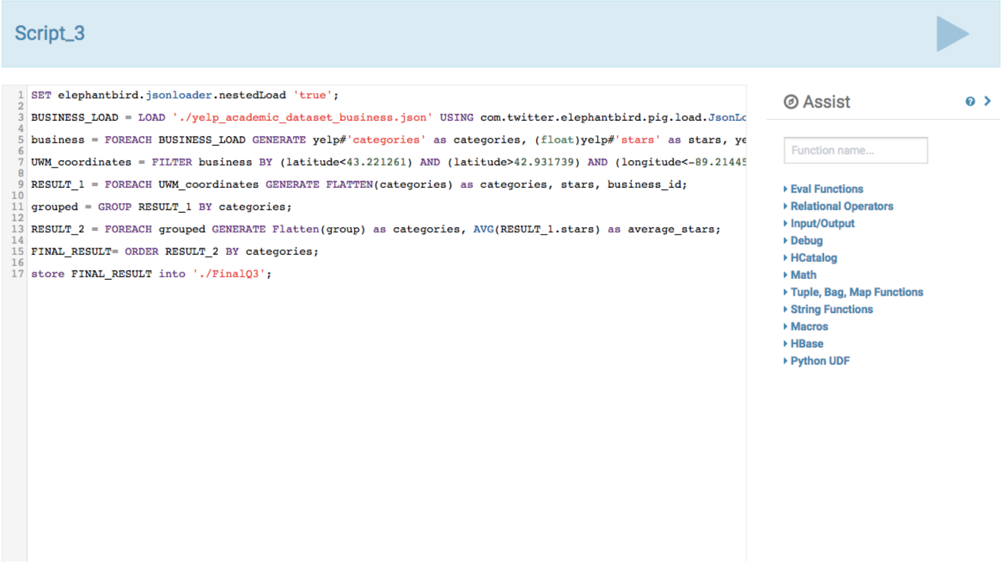
```
Surprise        Session Photography     5.0
Surprise        Personal Injury Law     5.0
Surprise        Home Window Tinting     5.0
Surprise        Carpet Installation     5.0
Broadlands      Contractors     5.0
Richmond Heights        Nightlife       5.0
Broadlands      Shopping        5.0
Longueuil       Coffee & Tea    5.0
Phoenix Kombucha        5.0
Scottsdale      IP & Internet Law       5.0
Surprise        Shredding Services      5.0
Pittsburgh      Bus Tours       5.0
Surprise        Pet Transportation      5.0
Surprise        Montessori Schools      5.0
Surprise        Mobile Dent Repair      5.0
Surprise        General Litigation      5.0
Surprise        Fireplace Services      5.0
Surprise        Damage Restoration      5.0
Surprise        Security Services       5.0
Monroeville     Hindu Temples   5.0
Surprise        Local Fish Stores       5.0
Fitchburg       Graphic Design  5.0
Surprise        Keys & Locksmiths       5.0
Surprise        Event Photography       5.0
Avalon  Audiologist     5.0
North Olmsted   Keys & Locksmiths       5.0
Surprise        Aquarium Services       5.0
```

Figure 5. Data in Output File 'FinalQ2'

3. What is the average rank (# stars) for businesses within 10 miles of the University of Wisconsin - Madison, by type of business?

   Center: University of Wisconsin - Madison
   Latitude: 43 04’ 30” N, Longitude: 89 25’ 2” W
   Decimal Degrees: Latitude: 43.0766, Longitude: -89.4125

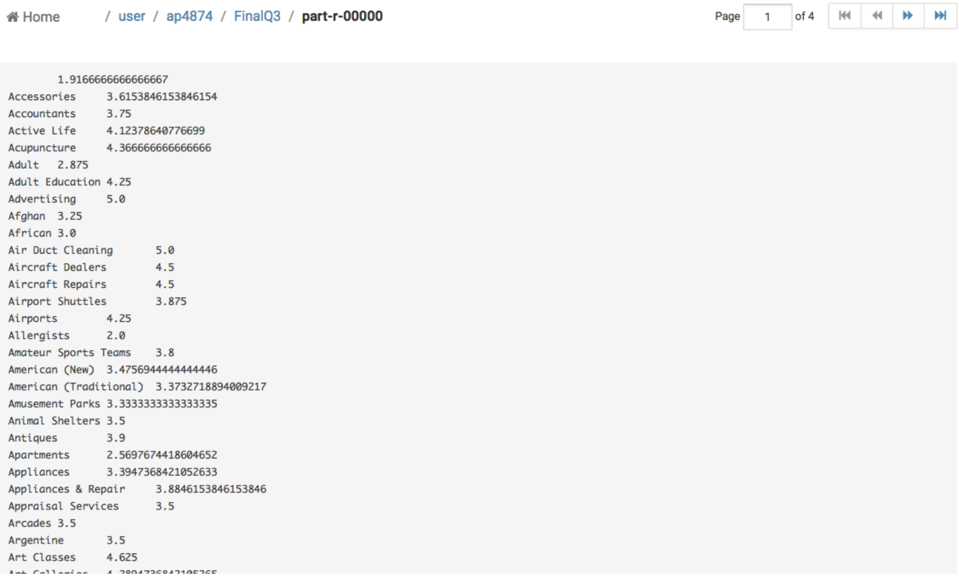
   The bounding box for this problem is ~10 miles, which we will loosely define as 10 minutes. So, the bounding box is a square box, 20 minutes long each side (of longitude and latitude), with UWM at the center.



```
SET elephantbird.jsonloader.nestedLoad 'true';

BUSINESS_LOAD = LOAD './yelp_academic_dataset_business.json' USING com.twitter.elephantbird.pig.load.JsonLc

business = FOREACH BUSINESS_LOAD GENERATE yelp#'categories' as categories, (float)yelp#'stars' as stars, ye

UWM_coordinates = FILTER business BY (latitude<43.221261) AND (latitude>42.931739) AND (longitude<-89.21445

RESULT_1 = FOREACH UWM_coordinates GENERATE FLATTEN(categories) as categories, stars, business_id;

grouped = GROUP RESULT_1 BY categories;

RESULT_2 = FOREACH grouped GENERATE Flatten(group) as categories, AVG(RESULT_1.stars) as average_stars;

FINAL_RESULT= ORDER RESULT_2 BY categories;

store FINAL_RESULT into './FinalQ3';
```

Figure 6. Pig Script

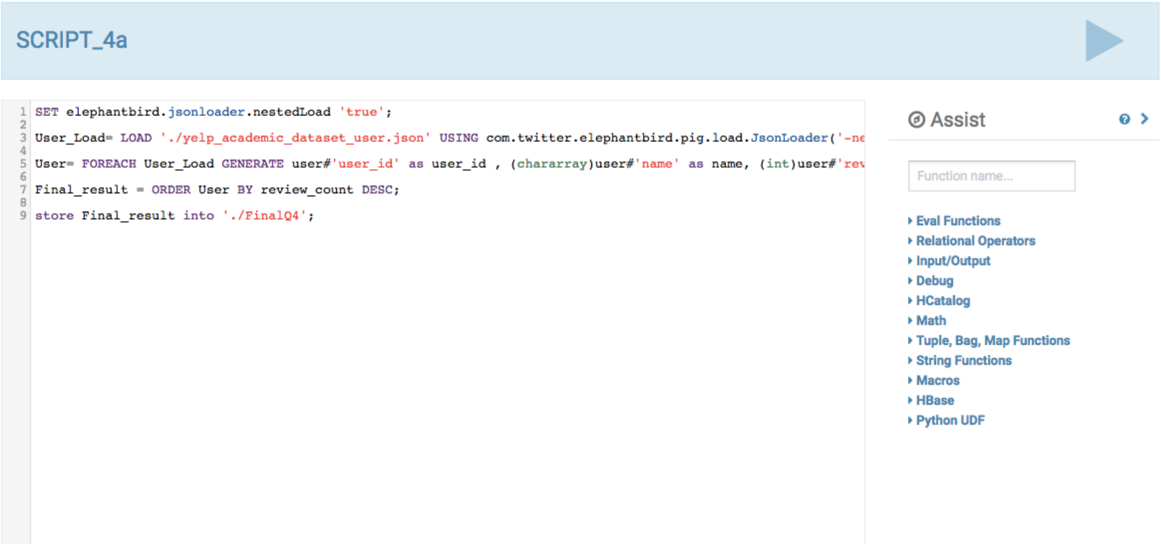
Explanation: To solve this query, first we need to get the coordinates around the UWM by considering UWM as center point. Imagine a square and get the coordinates (latitude, longitude) of the left, right, top and bottom of UWM with a distance of ~10 mile on each side. Then based on the coordinates filter out the rest of the area that falls outside the box (i.e. 10 miles on each side). Then I generated the columns that are required to solve the query and flatten the attribute ‘categories’. Then, performed the ‘group by’ operation on categories and calculate the average of # of stars using AVG (). Finally sorted the result based on categories and stored in ‘FinalQ3’.



```
	1.9166666666666667
Accessories	3.6153846153846154
Accountants	3.75
Active Life	4.12378640776699
Acupuncture	4.366666666666666
Adult	2.875
Adult Education	4.25
Advertising	5.0
Afghan	3.25
African	3.0
Air Duct Cleaning	5.0
Aircraft Dealers	4.5
Aircraft Repairs	4.5
Airport Shuttles	3.875
Airports	4.25
Allergists	2.0
Amateur Sports Teams	3.8
American (New)	3.4756944444444446
American (Traditional)	3.3732718894009217
Amusement Parks	3.3333333333333335
Animal Shelters	3.5
Antiques	3.9
Apartments	2.5697674418604652
Appliances	3.3947368421052633
Appliances & Repair	3.8846153846153846
Appraisal Services	3.5
Arcades	3.5
Argentine	3.5
Art Classes	4.625
Art Galleries	4.2894736842105265
```

Figure 7. Data in Output File 'FinalQ3'

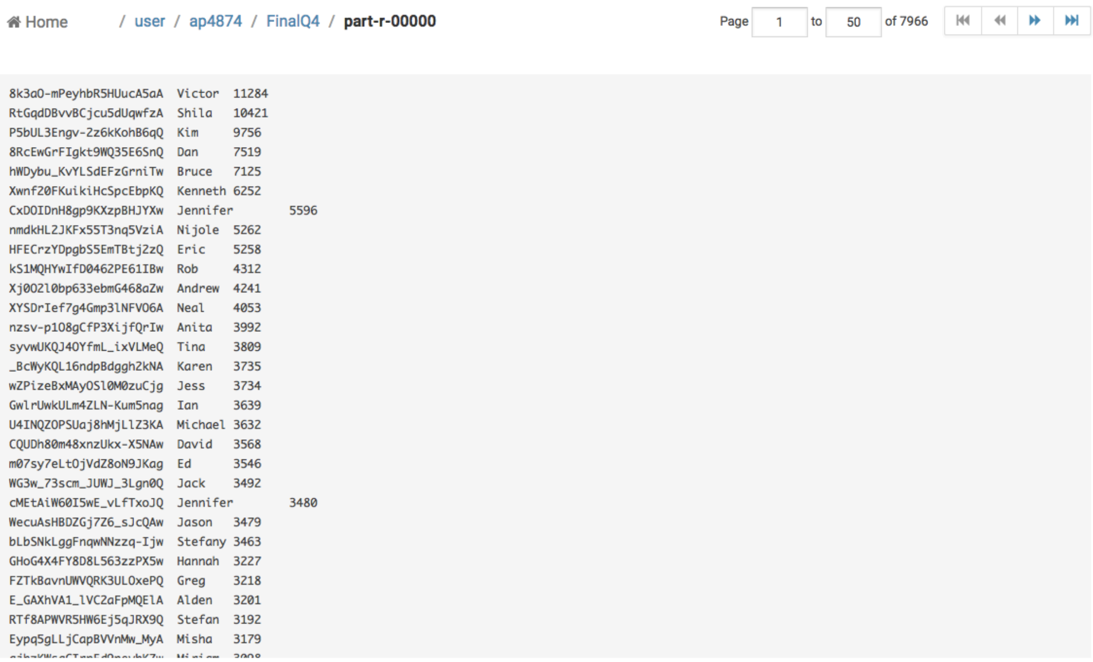
4. Rank reviewers by number of reviews. For the top 10 reviewers, show their average number of stars, by category.



```
SET elephantbird.jsonloader.nestedLoad 'true';

User_Load= LOAD './yelp_academic_dataset_user.json' USING com.twitter.elephantbird.pig.load.JsonLoader('-ne

User= FOREACH User_Load GENERATE user#'user_id' as user_id , (chararray)user#'name' as name, (int)user#'rev

Final_result = ORDER User BY review_count DESC;

store Final_result into './FinalQ4';
```

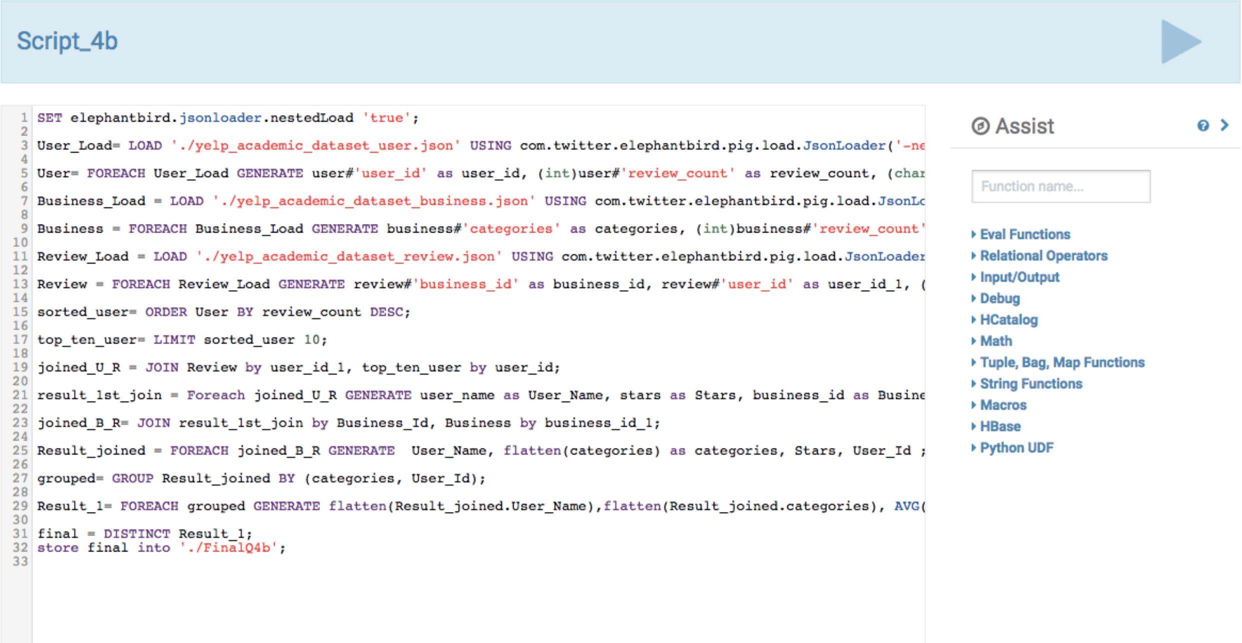
Figure 8. Pig Script

Explanation: Here we need to rank the reviewers based on number of reviews. For this I loaded the 'yelp_academic_dataset_user.json' file into the PIG and generate columns like 'User_id', 'Name' and 'review_count' from the relation 'User'. Then finally sort the result based on review_count in descending order and store the result in FinalQ4.



```
8k3aO-mPeyhbR5HUucA5aA  Victor  11284
RtGqdDBvvBCjcu5dUqwfzA  Shila   10421
P5bUL3Engv-2z6kKohB6qQ  Kim     9756
8RcEwGrFIgkt9WQ35E6SnQ  Dan     7519
hWDybu_KvYLSdEFzGrniTw  Bruce   7125
XwnfZ0FKuikiHcSpcEbpKQ  Kenneth 6252
CxDOIDnH8gp9KXzpBHJYXw  Jennifer        5596
nmdkHLZJKFx55T3nq5VziA  Nijole  5262
HFECrzYDpgbS5EmTBtjZzQ  Eric    5258
kS1MQHYwIfD0462PE61IBw  Rob     4312
Xj0O2l0bp633ebmG468aZw  Andrew  4241
XYSDrIef7g4Gmp3lNFVO6A  Neal    4053
nzsv-p1O8gCfP3XijfQrIw  Anita   3992
syvwUKQJ4OYfmL_ixVLMeQ  Tina    3809
_BcWyKQL16ndpBdggh2kNA  Karen   3735
wZPizeBxMAyOSl0M0zuCjg  Jess    3734
GwlrUwkULm4ZLN-Kum5nag  Ian     3639
U4INQZOPSUaj8hMjLlZ3KA  Michael 3632
CQUDh80m48xnzUkx-X5NAw  David   3568
m07sy7eLtOjVdZ8oN9JKag  Ed      3546
WG3w_73scm_JUWJ_3Lgn0Q  Jack    3492
cMEtAiW60I5wE_vLfTxoJQ  Jennifer        3480
WecuAsHBDZGj7Z6_sJcQAw  Jason   3479
bLbSNkLggFnqwNNzzq-Ijw  Stefany 3463
GHoG4X4FY8D8L563zzPX5w  Hannah  3227
FZTkBavnUWVQRK3ULOxePQ  Greg    3218
E_GAXhVA1_lVC2aFpMQElA  Alden   3201
RTf8APWVR5HW6Ej5qJRX9Q  Stefan  3192
Eypq5gLLjCapBVVnMw_MyA  Misha   3179
```

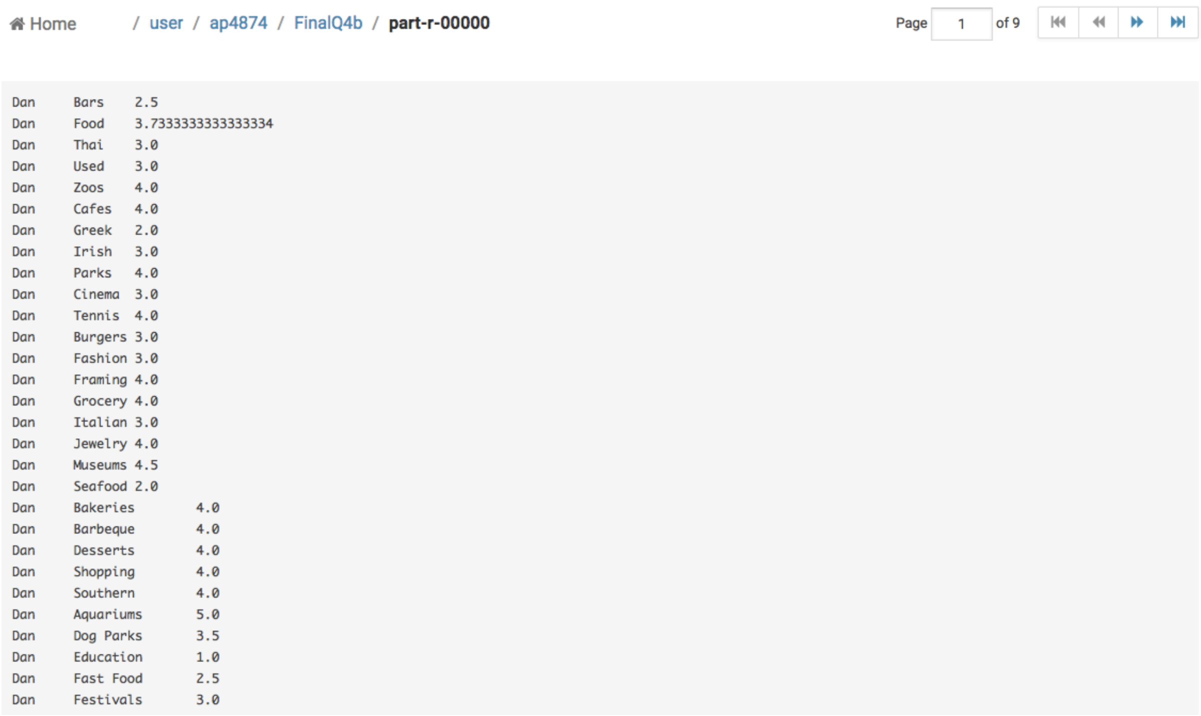
Figure 9. Data in Output File 'FinalQ4'



```
SET elephantbird.jsonloader.nestedLoad 'true';

User_Load= LOAD './yelp_academic_dataset_user.json' USING com.twitter.elephantbird.pig.load.JsonLoader('-ne

User= FOREACH User_Load GENERATE user#'user_id' as user_id, (int)user#'review_count' as review_count, (char

Business_Load = LOAD './yelp_academic_dataset_business.json' USING com.twitter.elephantbird.pig.load.JsonLc

Business = FOREACH Business_Load GENERATE business#'categories' as categories, (int)business#'review_count'

Review_Load = LOAD './yelp_academic_dataset_review.json' USING com.twitter.elephantbird.pig.load.JsonLoader

Review = FOREACH Review_Load GENERATE review#'business_id' as business_id, review#'user_id' as user_id_1, (

sorted_user= ORDER User BY review_count DESC;

top_ten_user= LIMIT sorted_user 10;

joined_U_R = JOIN Review by user_id_1, top_ten_user by user_id;

result_1st_join = Foreach joined_U_R GENERATE user_name as User_Name, stars as Stars, business_id as Busine

joined_B_R= JOIN result_1st_join by Business_Id, Business by business_id_1;

Result_joined = FOREACH joined_B_R GENERATE  User_Name, flatten(categories) as categories, Stars, User_Id ;

grouped= GROUP Result_joined BY (categories, User_Id);

Result_1= FOREACH grouped GENERATE flatten(Result_joined.User_Name),flatten(Result_joined.categories), AVG(

final = DISTINCT Result_1;
store final into './FinalQ4b';
```

Figure 10. Pig Script

To find out the average number of stars of the top 10 reviewers based on category, I also load the data from 'business.json' and 'review_json'. First, I performed the 'order by' operation on review_count under User to sort the data in descending order. This is done to get the Top 10 users. Then after doing this, used 'LIMIT' keyword to fetch the top 10 users based on review_count only. Then I have joined the User and Review table based on the attribute 'User_id'. From this join, I fetched detail like 'user_id', 'user_name', 'stars' and 'business_id' for those users that are common in both tables (User and Reviews). Again, performed join operation on the Business table and table generated from Ist join. As the data type of attribute 'categories' is complex, so I have used 'Flatten' keyword to simplified the data. Now, I have the number of stars given by top 10 users to particular category of different businesses. To calculate the average number of stars, I used AVG() on number of stars grouped by categories and user_id. Then get the distinct tuple using DISTINCT keyword and stored the result in FinalQ4b.



```
Dan     Bars    2.5
Dan     Food    3.7333333333333334
Dan     Thai    3.0
Dan     Used    3.0
Dan     Zoos    4.0
Dan     Cafes   4.0
Dan     Greek   2.0
Dan     Irish   3.0
Dan     Parks   4.0
Dan     Cinema  3.0
Dan     Tennis  4.0
Dan     Burgers 3.0
Dan     Fashion 3.0
Dan     Framing 4.0
Dan     Grocery 4.0
Dan     Italian 3.0
Dan     Jewelry 4.0
Dan     Museums 4.5
Dan     Seafood 2.0
Dan     Bakeries        4.0
Dan     Barbeque        4.0
Dan     Desserts        4.0
Dan     Shopping        4.0
Dan     Southern        4.0
Dan     Aquariums       5.0
Dan     Dog Parks       3.5
Dan     Education       1.0
Dan     Fast Food       2.5
Dan     Festivals       3.0
```

Figure 10. Data in Output File 'FinalQ4b'