[Big Data and Hadoop for Beginners - with Hands-on!](https://www.udemy.com/course/big-data-and-hadoop-for-beginners/)

Leave a rating

Your progress

Share

Practice-2: Designing Hive UDF

As we know User Defined Functions in Hive, also known as UDF, allow you to create custom functions to process records or groups of records. For example, a UDF could perform calculations using an external math library, combine several column values into one, do geospatial calculations, or other kinds of tests and transformations that are outside the scope of the built-in SQL operators and functions.

I hope you enjoyed doing exercise on ***Analyzing Taxi Trips Data***. If you remember, in **Taxi Trips** raw data, we saw there are pick-up and drop-off latitude and longitude for each taxi trips. How about If I ask you to group all lat-long that belongs to Midtown, and show me the trends on pickup and drop-offs in that area.



Well, there can be a lot of ways to solve this problem; reading some Map API to generate area names from latitude and longitude or generating Geohash of given latitude and longitude and then solving your problems further.   
  
In this exercise we will write a **UDF in Hive** to generate **Geohash**, and using that UDF we will solve this problem.

First of all, let’s understand what is Geohash.

What is **Geohash**?

A Geohash is a convenient way of expressing a location, anywhere in the world, using a short alphanumeric string, with greater precision obtained with longer strings. It converts given latitude and longitude into a string representation.  
  
E.g. Geohash(lat,long)  
Geohash(-73.9832763672,40.7138175964) --> hfugn7

If you have a group of latitude and longitude belonging to an area, and when you encode them, you will see the initial characters of the string will be similar. Below is an example:  
  
-73.9832763672, 40.7138175964, hfugn7

-73.9938964844, 40.7501106262, hfugp3

-74.0016479492, 40.7242431641, hfugn8

-74.0090866089, 40.7138175964, hfufyr

-73.9711761475, 40.7624282837, hfugpt

-73.8743743896, 40.7740478516, hfugxz

-73.9832763672, 40.7260093689, hfugne

-74.0026626587, 40.7341423035, hfugnb

-73.7830429077, 40.6443557739, hfuukb

-73.9817886353, 40.765335083, hfugps

-73.9844207764, 40.748260498, hfugp5

-73.9942398071, 40.7508049011, hfugp3

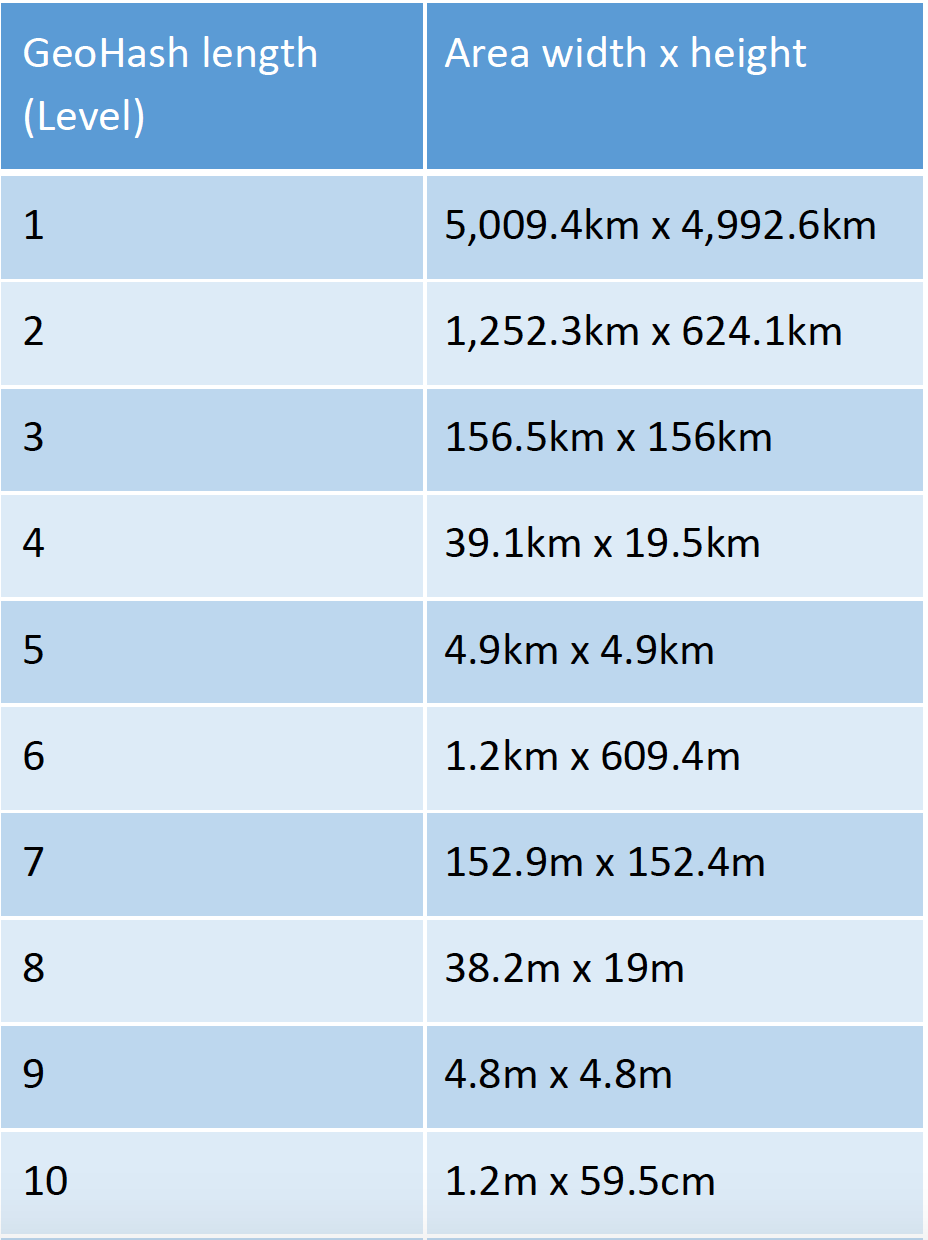
-73.9808273315, 40.7445297241, hfugph

-73.9827423096, 40.7281837463, hfugng

-73.9793243408, 40.7495498657, hfugpk

-73.9603500366, 40.7663993835, hfugpx

A Geohash can be represented at various levels. Above examples and in this task we will be using Geohash Level 6 (Six characters string) which is hard-coded in our code. Below is a chart that shows how much area is covered at various Geohash levels:



Benefits of Geohash:  
\* It can be used as a unique Identifier   
\* It can be used to store point data into Database  
\* Point Lookup can be faster because of storing as a string and making Index on it.  
\* Can be a good use case in Geo Data Analysis.  
  
Wiki [Geohash](https://en.wikipedia.org/wiki/Geohash" \t "_blank)  
  
Below is the Python code that generates Geohash. I won’t go deep into this code as I am assuming you have basics knowledge of Python.

[Python Code for UDF]

1. from math import log10
2. import sys
3. import sys
4. \_\_base32 = '0123456789bcdefghjkmnpqrstuvwxyz'
5. \_\_decodemap = { }
6. for i in range(len(\_\_base32)):
7. \_\_decodemap[\_\_base32[i]] = i
8. del i
9. def encode(latitude, longitude, precision=6):
10. """
11. Encode a position given in float arguments latitude, longitude to
12. a geohash which will have the character count precision.
13. """
14. lat\_interval, lon\_interval = (-90.0, 90.0), (-180.0, 180.0)
15. geohash = []
16. bits = [ 16, 8, 4, 2, 1 ]
17. bit = 0
18. ch = 0
19. even = True
20. while len(geohash) < precision:
21. if even:
22. mid = (lon\_interval[0] + lon\_interval[1]) / 2
23. if longitude > mid:
24. ch |= bits[bit]
25. lon\_interval = (mid, lon\_interval[1])
26. else:
27. lon\_interval = (lon\_interval[0], mid)
28. else:
29. mid = (lat\_interval[0] + lat\_interval[1]) / 2
30. if latitude > mid:
31. ch |= bits[bit]
32. lat\_interval = (mid, lat\_interval[1])
33. else:
34. lat\_interval = (lat\_interval[0], mid)
35. even = not even
36. if bit < 4:
37. bit += 1
38. else:
39. geohash += \_\_base32[ch]
40. bit = 0
41. ch = 0
42. return ''.join(geohash)
43. for line in sys.stdin:
44. tpep\_pickup\_datetime,tpep\_dropoff\_datetime,passenger\_count,trip\_distance,total\_amount,lat,lon,code = line.strip().split('\t')
45. lat=float(lat)
46. lon=float(lon)
47. code=encode(lat,lon)
48. print ','.join([str(tpep\_pickup\_datetime),str(tpep\_dropoff\_datetime),str(passenger\_count),str(trip\_distance),str(total\_amount),str(lat),str(lon),str(code)])

**Now follow below steps to solve this problem:**

**1. Start your VM and login to HDP Sandbox.**

     ssh root@localhost -p2222

**2. Download Taxi Trips raw file from:**

http://www.nyc.gov/html/tlc/html/about/trip\_record\_data.shtml

I would recommend download 2015 Yellow taxi data for this exercise. Once you login to the terminal, below is how you can download data:

1. wget https://s3.amazonaws.com/nyc-tlc/trip+data/yellow\_tripdata\_2015-01.csv
2. :
3. :
4. wget https://s3.amazonaws.com/nyc-tlc/trip+data/yellow\_tripdata\_2015-12.csv

**3. Run below commands from terminal:**

1. sudo su - hdfs
2. hdfs dfs -mkdir /user/root
3. hdfs dfs -mkdir /user/nyc\_taxi
4. hdfs dfs -chown root:hdfs /user/root
5. hdfs dfs -chown root:hdfs /user/nyc\_taxi
6. exit
7. hadoop fs -put yellow\_tripdata\_\*.csv /user/nyc\_taxi/

**4. Create table to read raw data:**

1. sudo hive
3. hive>
4. CREATE EXTERNAL TABLE yellow\_trips (
5. vendorid string,
6. tpep\_pickup\_datetime string,
7. tpep\_dropoff\_datetime string,
8. passenger\_count string,
9. trip\_distance string,
10. pickup\_longitude double,
11. pickup\_latitude double,
12. ratecodeid string,
13. store\_and\_fwd\_flag string,
14. dropoff\_longitude double,
15. dropoff\_latitude double,
16. payment\_type string,
17. fare\_amount string,
18. extra string,
19. mta\_tax string,
20. tip\_amount string,
21. tolls\_amount string,
22. improvement\_surcharge string,
23. total\_amount string )
24. ROW FORMAT DELIMITED
25. FIELDS TERMINATED BY ','
26. STORED AS TEXTFILE
27. location '/user/nyc\_taxi/'
28. tblproperties ("skip.header.line.count"="1");

**5. Download attached python file (attached in this lecture), and copy to terminal:**

scp -P 2222 ~/Downloads/geohash\_udf.py root@localhost:/root/

**6. Login to Hive Shell:**

1. sudo hive
3. then run below command from Hive shell:
5. hive> add file /root/geohash\_udf.py;
7. Now we have added the file, now below is how we will be running UDF in Hive:
9. hive> set hive.execution.engine=tez;

Running SQL using Python UDF:

1. hive>
2. SELECT
3. TRANSFORM (tpep\_pickup\_datetime,
4. tpep\_dropoff\_datetime,
5. passenger\_count,
6. trip\_distance,
7. total\_amount,
8. pickup\_latitude,
9. pickup\_longitude,
10. code) USING 'python geohash\_udf.py' AS tpep\_pickup\_datetime,
11. tpep\_dropoff\_datetime,
12. passenger\_count,
13. trip\_distance,
14. total\_amount,
15. pickup\_latitude,
16. pickup\_longitude,
17. code
18. FROM
19. (SELECT tpep\_pickup\_datetime,
20. tpep\_dropoff\_datetime,
21. passenger\_count,
22. trip\_distance,
23. total\_amount,
24. pickup\_latitude,
25. pickup\_longitude,
26. 'txt' AS code
27. FROM yellow\_trips limit 10) a;

**7. Create table to store processed data:**

1. create table yellow\_trips\_processed
2. (
3. tpep\_pickup\_datetime string,
4. tpep\_dropoff\_datetime string,
5. passenger\_count string,
6. trip\_distance string,
7. total\_amount string,
8. pickup\_latitude string,
9. pickup\_longitude string,
10. geohash string
11. )
12. ROW FORMAT DELIMITED
13. FIELDS TERMINATED BY ','
14. STORED AS TEXTFILE ;

**8. Process data and load into above table:**

1. hive> set hive.execution.engine=tez;
3. hive> insert overwrite table yellow\_trips\_processed
4. select
5. TRANSFORM (tpep\_pickup\_datetime,tpep\_dropoff\_datetime,passenger\_count,trip\_distance,total\_amount,pickup\_latitude, pickup\_longitude,code)
6. using 'python geohash\_udf.py' as tpep\_pickup\_datetime,tpep\_dropoff\_datetime,passenger\_count,trip\_distance,total\_amount,pickup\_latitude, pickup\_longitude,code
7. from
8. (
9. select tpep\_pickup\_datetime,tpep\_dropoff\_datetime,passenger\_count,trip\_distance,total\_amount,pickup\_latitude,pickup\_longitude,'txt' as code from yellow\_trips
10. ) a;

**9. Midtown data analysis.**  
  
**Midtown Lat-Long: 40.758896, -73.985130  
Midtown Geohash: dr5ru7**

1. --Doing group by on Date and Hours to understand the ride pattern in Midtown area
3. SELECT date(cast(tpep\_pickup\_datetime AS TIMESTAMP)) AS pickup\_date,
4. hour(cast(tpep\_pickup\_datetime AS TIMESTAMP)) AS pickup\_hour,
5. count(1) AS c\_rides
6. FROM yellow\_trips\_processed
7. WHERE geohash LIKE 'dr5ru%' --looking data for Midtown
8. GROUP BY date(cast(tpep\_pickup\_datetime AS TIMESTAMP)),
9. hour(cast(tpep\_pickup\_datetime AS TIMESTAMP)) ;