INFORMATICS INSTITUTE OF TECHNOLOGY

In Collaboration with

ROBERT GORDON UNIVERSITY ABERDEEN

School of Computing Science and Digital Media

MSc Big Data Analytics

2019/2020

By

Niroshan Kumarasinghe

IIT No: 2019269

RGU No: 1912833

CMM705 – Big Data Programming

Coursework

1. Part One - Deployment Architecture

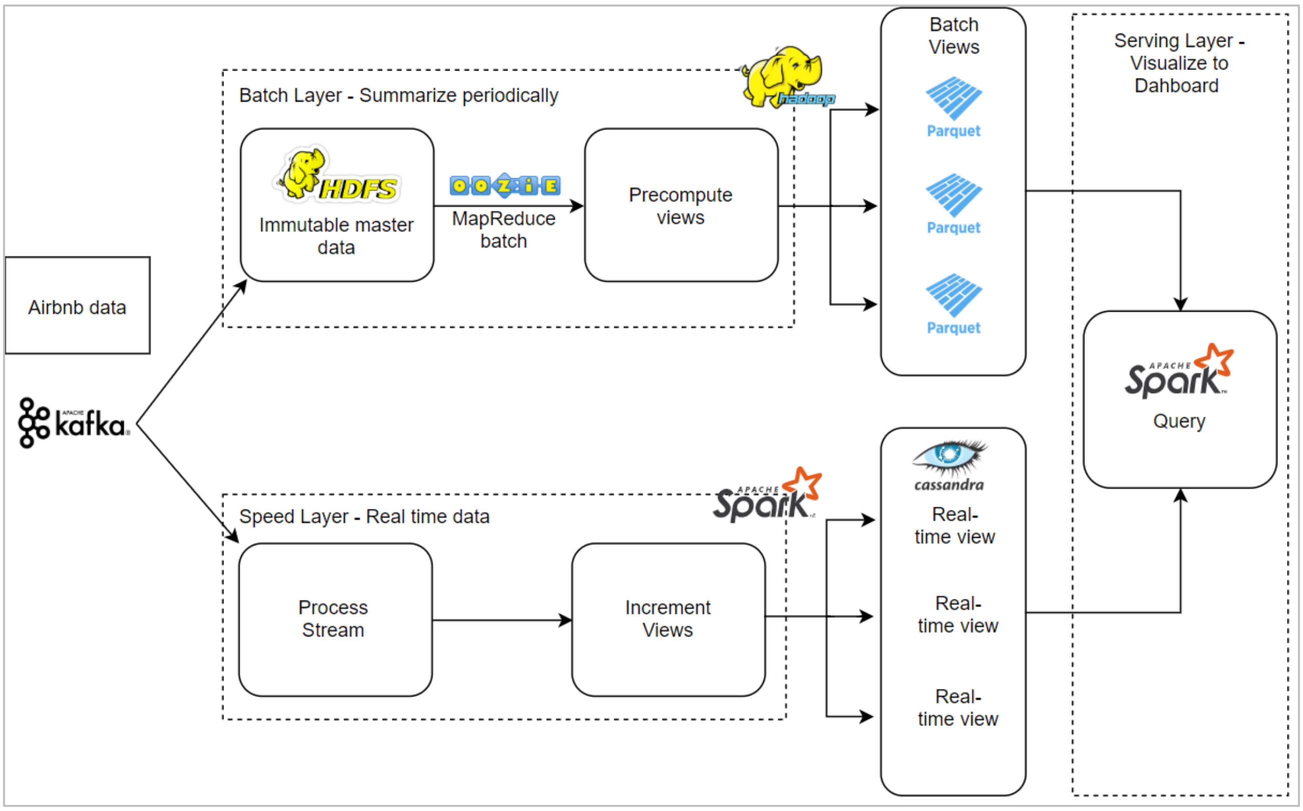


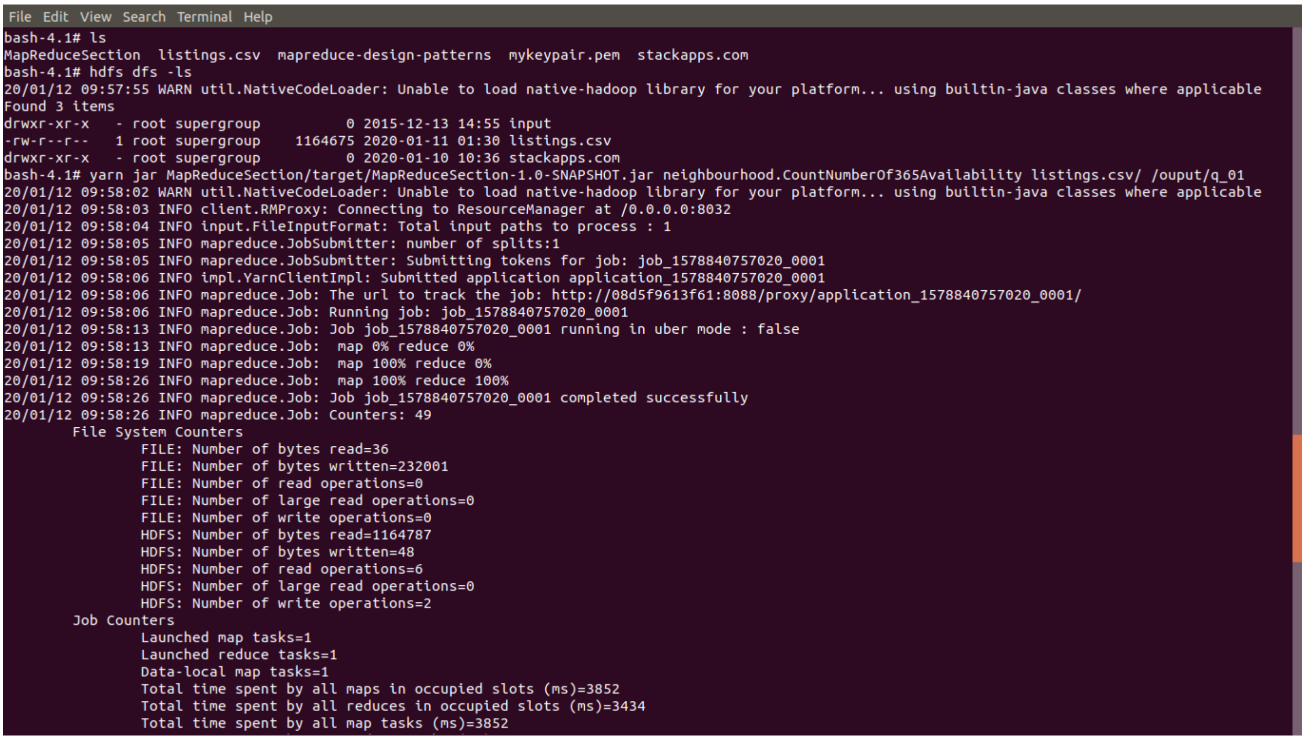
Figure 1: Deployment Diagram (Question 01)

Tool	Implementation
Kafka	Feeding data from Airbnb data sources
HDFS	Store immutable master data (archive)
Oozie	Manage data and Schedule workflow
Spark	Process streams for real-time views
Cassandra	Store real-time view (hot path)
Parquet	Create batch view and store
Spark	Serve to dashboard

2. Map reduce jobs/queries and results

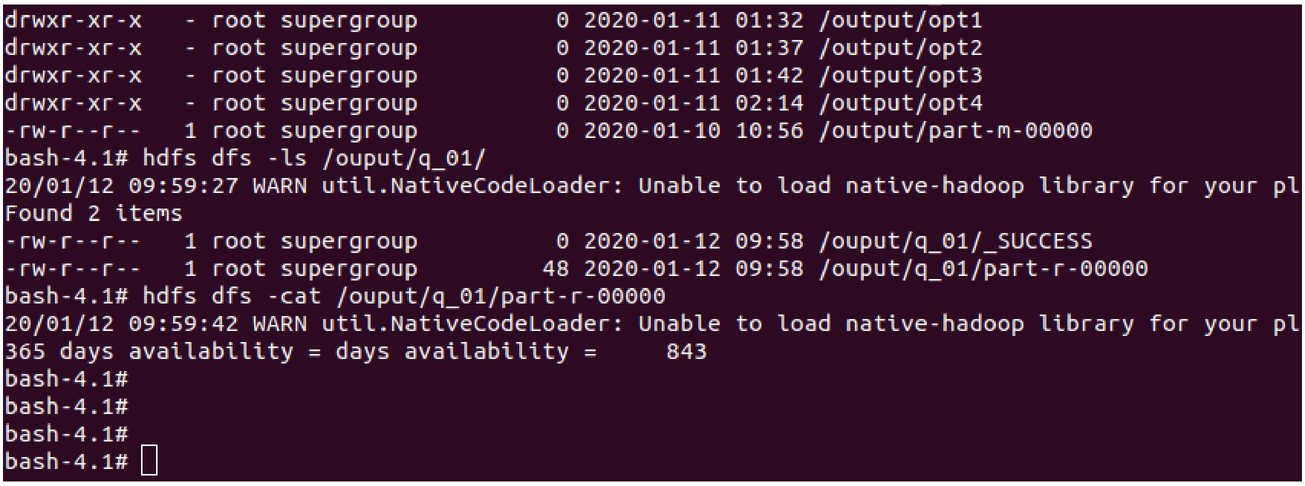
2.1. Hadoop Map Reduce

Question 01 – Count of 356 Availability Rentals



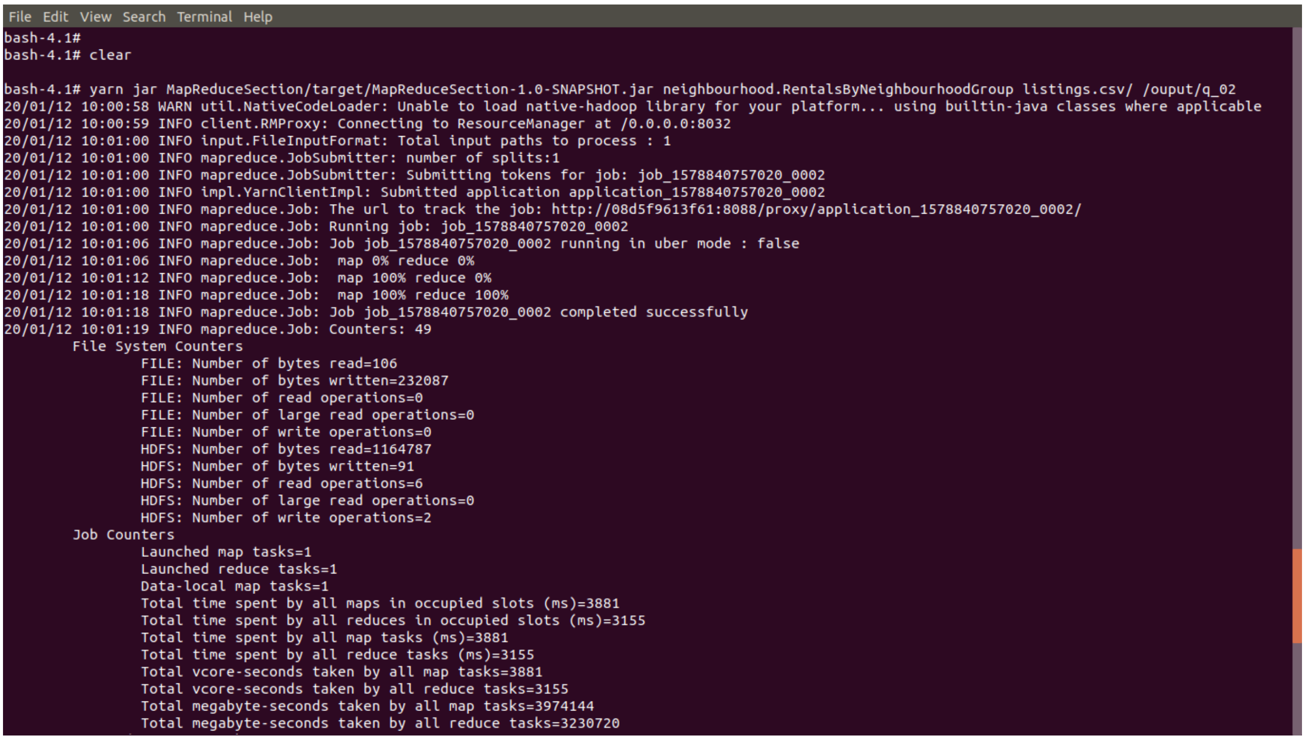
```
bash-4.1# ls
MapReduceSection  listings.csv  mapreduce-design-patterns  mykeypair.pem  stackapps.com
bash-4.1# hdfs dfs -ls
20/01/12 09:57:55 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 3 items
drwxr-xr-x   - root supergroup          0 2015-12-13 14:55 input
-rw-r--r--   1 root supergroup    1164675 2020-01-11 01:30 listings.csv
drwxr-xr-x   - root supergroup          0 2020-01-10 10:36 stackapps.com
bash-4.1# yarn jar MapReduceSection/target/MapReduceSection-1.0-SNAPSHOT.jar neighbourhood.CountNumberOf365Availability listings.csv/ /ouput/q_01
20/01/12 09:58:02 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
20/01/12 09:58:03 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
20/01/12 09:58:04 INFO input.FileInputFormat: Total input paths to process : 1
20/01/12 09:58:05 INFO mapreduce.JobSubmitter: number of splits:1
20/01/12 09:58:05 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1578840757020_0001
20/01/12 09:58:06 INFO impl.YarnClientImpl: Submitted application application_1578840757020_0001
20/01/12 09:58:06 INFO mapreduce.Job: The url to track the job: http://08d5f9613f61:8088/proxy/application_1578840757020_0001/
20/01/12 09:58:06 INFO mapreduce.Job: Running job: job_1578840757020_0001
20/01/12 09:58:13 INFO mapreduce.Job: Job job_1578840757020_0001 running in uber mode : false
20/01/12 09:58:13 INFO mapreduce.Job:  map 0% reduce 0%
20/01/12 09:58:19 INFO mapreduce.Job:  map 100% reduce 0%
20/01/12 09:58:26 INFO mapreduce.Job:  map 100% reduce 100%
20/01/12 09:58:26 INFO mapreduce.Job: Job job_1578840757020_0001 completed successfully
20/01/12 09:58:26 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=36
                FILE: Number of bytes written=232001
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1164787
                HDFS: Number of bytes written=48
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=3852
                Total time spent by all reduces in occupied slots (ms)=3434
                Total time spent by all map tasks (ms)=3852
```

Question 01: output



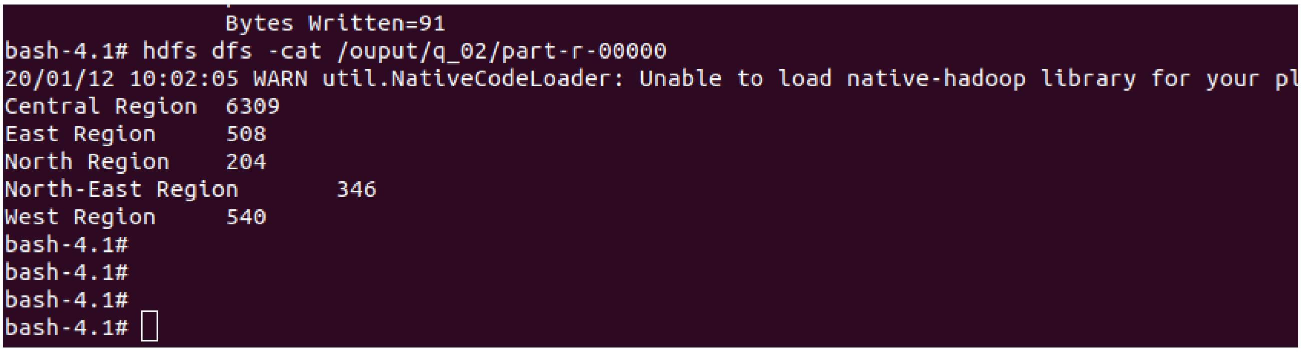
```
drwxr-xr-x   - root supergroup          0 2020-01-11 01:32 /output/opt1
drwxr-xr-x   - root supergroup          0 2020-01-11 01:37 /output/opt2
drwxr-xr-x   - root supergroup          0 2020-01-11 01:42 /output/opt3
drwxr-xr-x   - root supergroup          0 2020-01-11 02:14 /output/opt4
-rw-r--r--   1 root supergroup          0 2020-01-10 10:56 /output/part-m-00000
bash-4.1# hdfs dfs -ls /ouput/q_01/
20/01/12 09:59:27 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your pl
Found 2 items
-rw-r--r--   1 root supergroup          0 2020-01-12 09:58 /ouput/q_01/_SUCCESS
-rw-r--r--   1 root supergroup         48 2020-01-12 09:58 /ouput/q_01/part-r-00000
bash-4.1# hdfs dfs -cat /ouput/q_01/part-r-00000
20/01/12 09:59:42 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your pl
365 days availability = days availability =     843
bash-4.1#
bash-4.1#
bash-4.1#
bash-4.1# 
```

Question 02 – Group Rentals by Neighborhood Group



```
bash-4.1#
bash-4.1# clear

bash-4.1# yarn jar MapReduceSection/target/MapReduceSection-1.0-SNAPSHOT.jar neighbourhood.RentalsByNeighbourhoodGroup listings.csv/ /ouput/q_02
20/01/12 10:00:58 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
20/01/12 10:00:59 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
20/01/12 10:01:00 INFO input.FileInputFormat: Total input paths to process : 1
20/01/12 10:01:00 INFO mapreduce.JobSubmitter: number of splits:1
20/01/12 10:01:00 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1578840757020_0002
20/01/12 10:01:00 INFO impl.YarnClientImpl: Submitted application application_1578840757020_0002
20/01/12 10:01:00 INFO mapreduce.Job: The url to track the job: http://08d5f9613f61:8088/proxy/application_1578840757020_0002/
20/01/12 10:01:00 INFO mapreduce.Job: Running job: job_1578840757020_0002
20/01/12 10:01:06 INFO mapreduce.Job: Job job_1578840757020_0002 running in uber mode : false
20/01/12 10:01:06 INFO mapreduce.Job:  map 0% reduce 0%
20/01/12 10:01:12 INFO mapreduce.Job:  map 100% reduce 0%
20/01/12 10:01:18 INFO mapreduce.Job:  map 100% reduce 100%
20/01/12 10:01:18 INFO mapreduce.Job: Job job_1578840757020_0002 completed successfully
20/01/12 10:01:19 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=106
                FILE: Number of bytes written=232087
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1164787
                HDFS: Number of bytes written=91
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=3881
                Total time spent by all reduces in occupied slots (ms)=3155
                Total time spent by all map tasks (ms)=3881
                Total time spent by all reduce tasks (ms)=3155
                Total vcore-seconds taken by all map tasks=3881
                Total vcore-seconds taken by all reduce tasks=3155
                Total megabyte-seconds taken by all map tasks=3974144
                Total megabyte-seconds taken by all reduce tasks=3230720
```

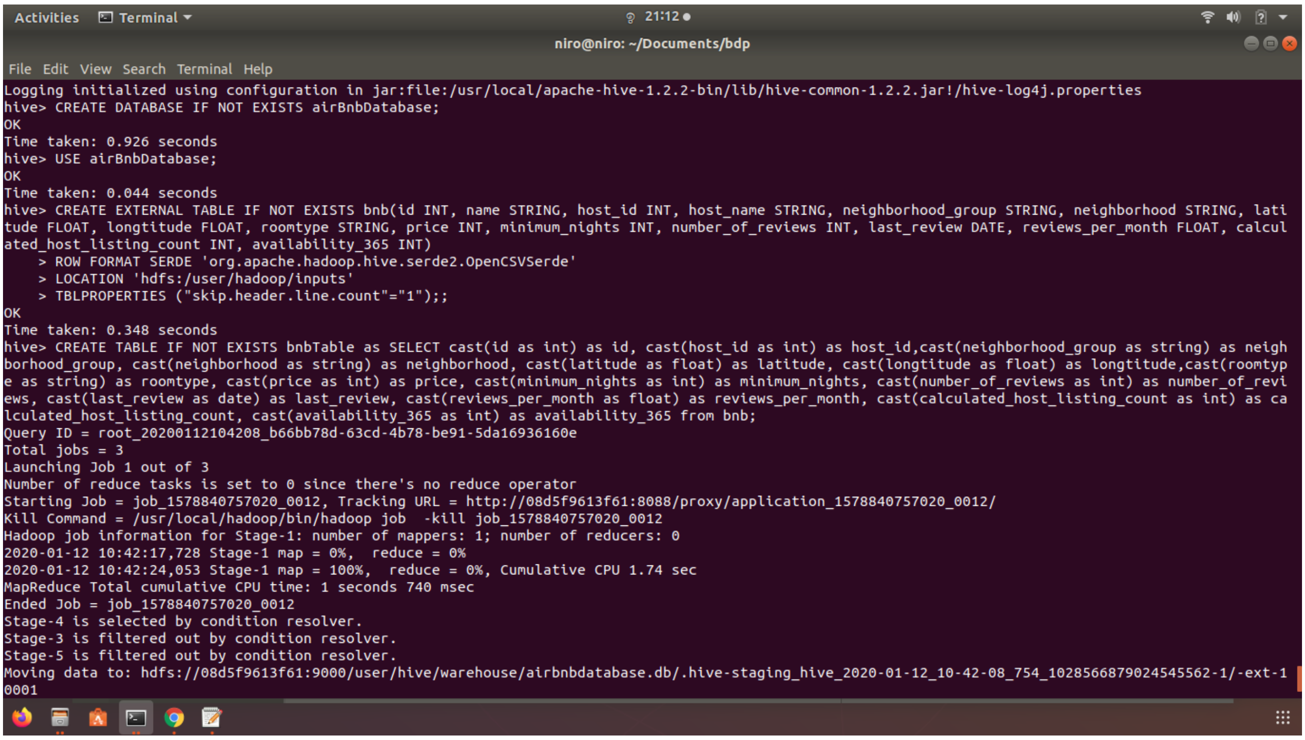
Question 02: output



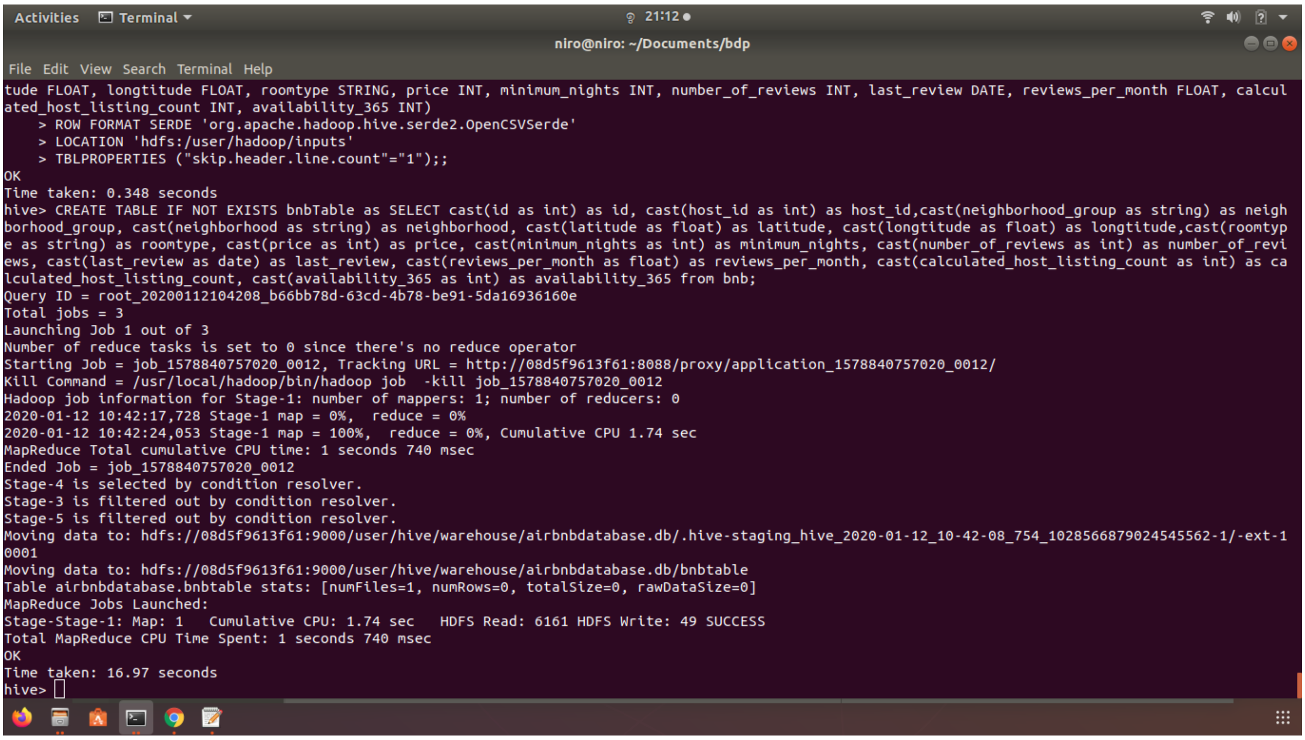
```
                Bytes Written=91
bash-4.1# hdfs dfs -cat /ouput/q_02/part-r-00000
20/01/12 10:02:05 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your pl
Central Region  6309
East Region     508
North Region    204
North-East Region       346
West Region     540
bash-4.1#
bash-4.1#
bash-4.1#
bash-4.1# 
```

2.2. Hive

Setup: Create databases, create tables, insert and convert data

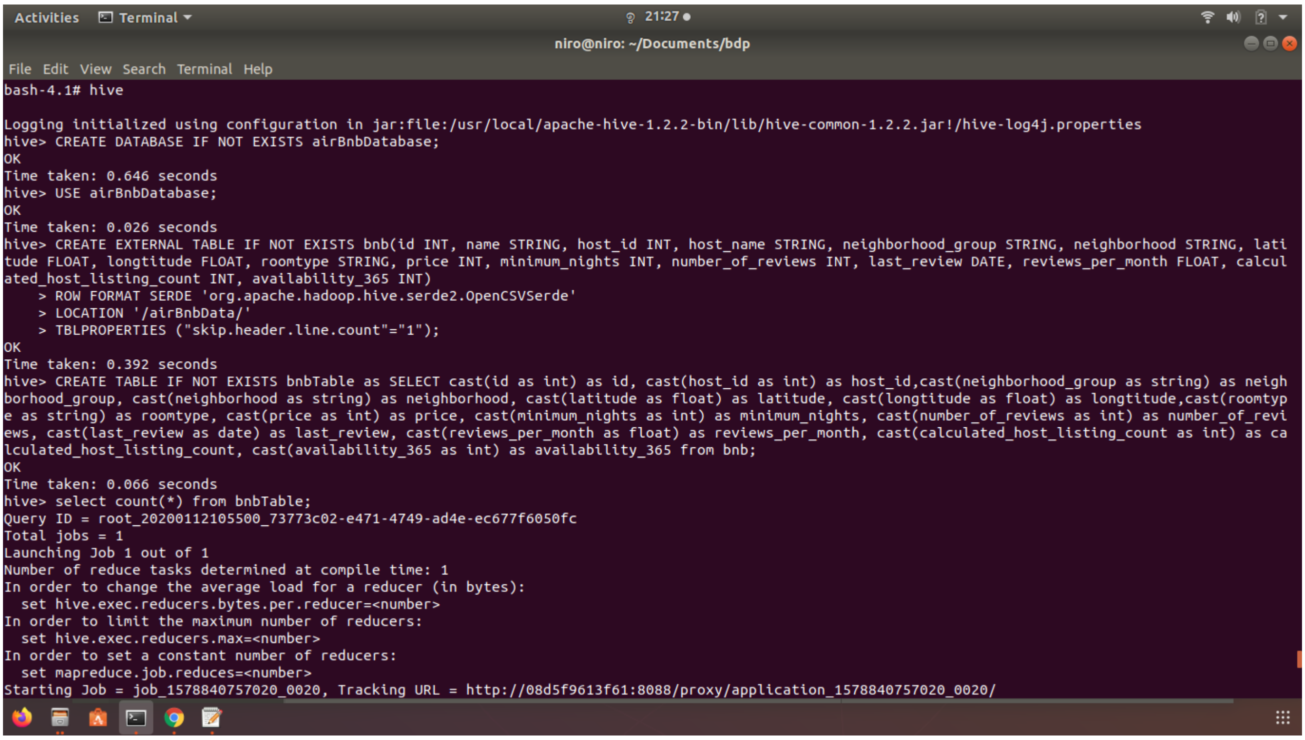

Activities
Terminal
21:12
niro@niro: ~/Documents/bdp
File Edit View Search Terminal Help

```
Logging initialized using configuration in jar:file:/usr/local/apache-hive-1.2.2-bin/lib/hive-common-1.2.2.jar!/hive-log4j.properties
hive> CREATE DATABASE IF NOT EXISTS airBnbDatabase;
OK
Time taken: 0.926 seconds
hive> USE airBnbDatabase;
OK
Time taken: 0.044 seconds
hive> CREATE EXTERNAL TABLE IF NOT EXISTS bnb(id INT, name STRING, host_id INT, host_name STRING, neighborhood_group STRING, neighborhood STRING, lati
tude FLOAT, longtitude FLOAT, roomtype STRING, price INT, minimum_nights INT, number_of_reviews INT, last_review DATE, reviews_per_month FLOAT, calcul
ated_host_listing_count INT, availability_365 INT)
    > ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
    > LOCATION 'hdfs:/user/hadoop/inputs'
    > TBLPROPERTIES ("skip.header.line.count"="1");;
OK
Time taken: 0.348 seconds
hive> CREATE TABLE IF NOT EXISTS bnbTable as SELECT cast(id as int) as id, cast(host_id as int) as host_id,cast(neighborhood_group as string) as neigh
borhood_group, cast(neighborhood as string) as neighborhood, cast(latitude as float) as latitude, cast(longtitude as float) as longtitude,cast(roomtyp
e as string) as roomtype, cast(price as int) as price, cast(minimum_nights as int) as minimum_nights, cast(number_of_reviews as int) as number_of_revi
ews, cast(last_review as date) as last_review, cast(reviews_per_month as float) as reviews_per_month, cast(calculated_host_listing_count as int) as ca
lculated_host_listing_count, cast(availability_365 as int) as availability_365 from bnb;
Query ID = root_20200112104208_b66bb78d-63cd-4b78-be91-5da16936160e
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1578840757020_0012, Tracking URL = http://08d5f9613f61:8088/proxy/application_1578840757020_0012/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1578840757020_0012
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2020-01-12 10:42:17,728 Stage-1 map = 0%,  reduce = 0%
2020-01-12 10:42:24,053 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.74 sec
MapReduce Total cumulative CPU time: 1 seconds 740 msec
Ended Job = job_1578840757020_0012
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://08d5f9613f61:9000/user/hive/warehouse/airbnbdatabase.db/.hive-staging_hive_2020-01-12_10-42-08_754_1028566879024545562-1/-ext-1
0001
```


Activities
Terminal
21:12
niro@niro: ~/Documents/bdp
File Edit View Search Terminal Help

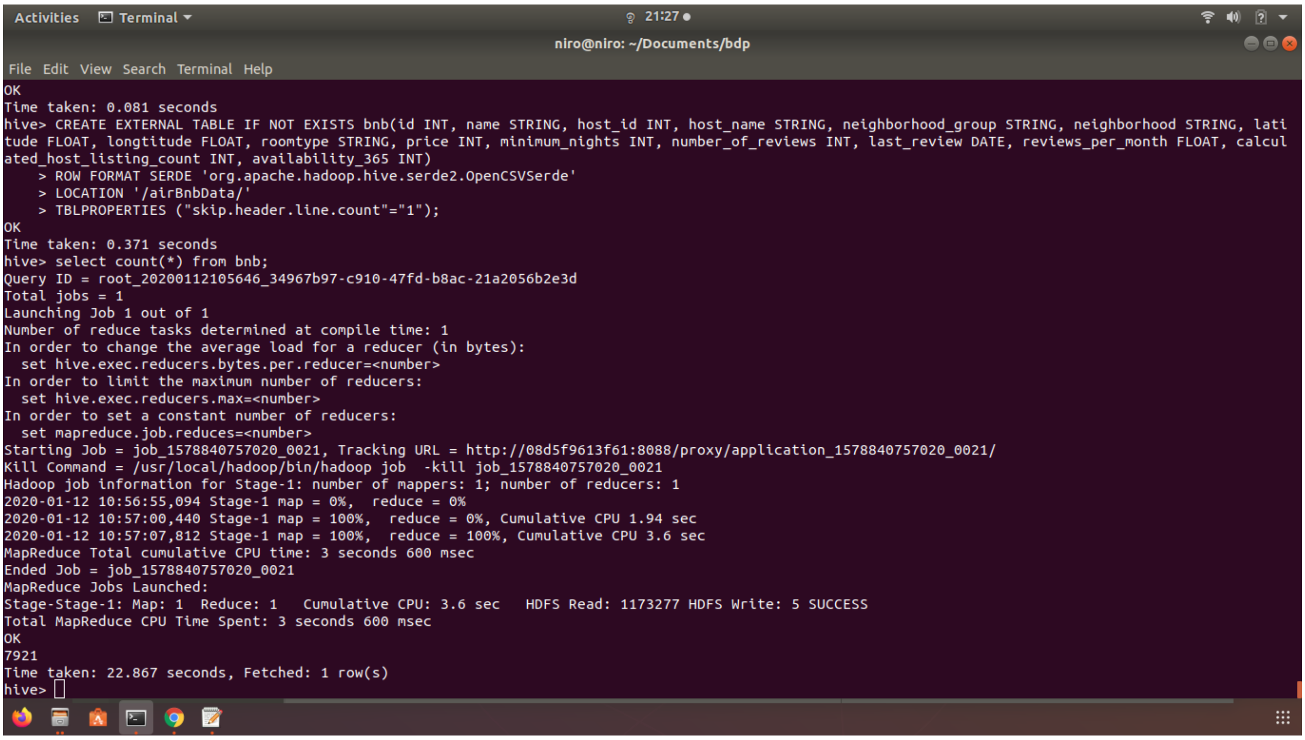
```
tude FLOAT, longtitude FLOAT, roomtype STRING, price INT, minimum_nights INT, number_of_reviews INT, last_review DATE, reviews_per_month FLOAT, calcul
ated_host_listing_count INT, availability_365 INT)
    > ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
    > LOCATION 'hdfs:/user/hadoop/inputs'
    > TBLPROPERTIES ("skip.header.line.count"="1");;
OK
Time taken: 0.348 seconds
hive> CREATE TABLE IF NOT EXISTS bnbTable as SELECT cast(id as int) as id, cast(host_id as int) as host_id,cast(neighborhood_group as string) as neigh
borhood_group, cast(neighborhood as string) as neighborhood, cast(latitude as float) as latitude, cast(longtitude as float) as longtitude,cast(roomtyp
e as string) as roomtype, cast(price as int) as price, cast(minimum_nights as int) as minimum_nights, cast(number_of_reviews as int) as number_of_revi
ews, cast(last_review as date) as last_review, cast(reviews_per_month as float) as reviews_per_month, cast(calculated_host_listing_count as int) as ca
lculated_host_listing_count, cast(availability_365 as int) as availability_365 from bnb;
Query ID = root_20200112104208_b66bb78d-63cd-4b78-be91-5da16936160e
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1578840757020_0012, Tracking URL = http://08d5f9613f61:8088/proxy/application_1578840757020_0012/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1578840757020_0012
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2020-01-12 10:42:17,728 Stage-1 map = 0%,  reduce = 0%
2020-01-12 10:42:24,053 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.74 sec
MapReduce Total cumulative CPU time: 1 seconds 740 msec
Ended Job = job_1578840757020_0012
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://08d5f9613f61:9000/user/hive/warehouse/airbnbdatabase.db/.hive-staging_hive_2020-01-12_10-42-08_754_1028566879024545562-1/-ext-1
0001
```

```
Moving data to: hdfs://08d5f9613f61:9000/user/hive/warehouse/airbnbdatabase.db/bnbtable
Table airbnbdatabase.bnbtable stats: [numFiles=1, numRows=0, totalSize=0, rawDataSize=0]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 1.74 sec   HDFS Read: 6161 HDFS Write: 49 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 740 msec
OK
Time taken: 16.97 seconds
hive> 
```



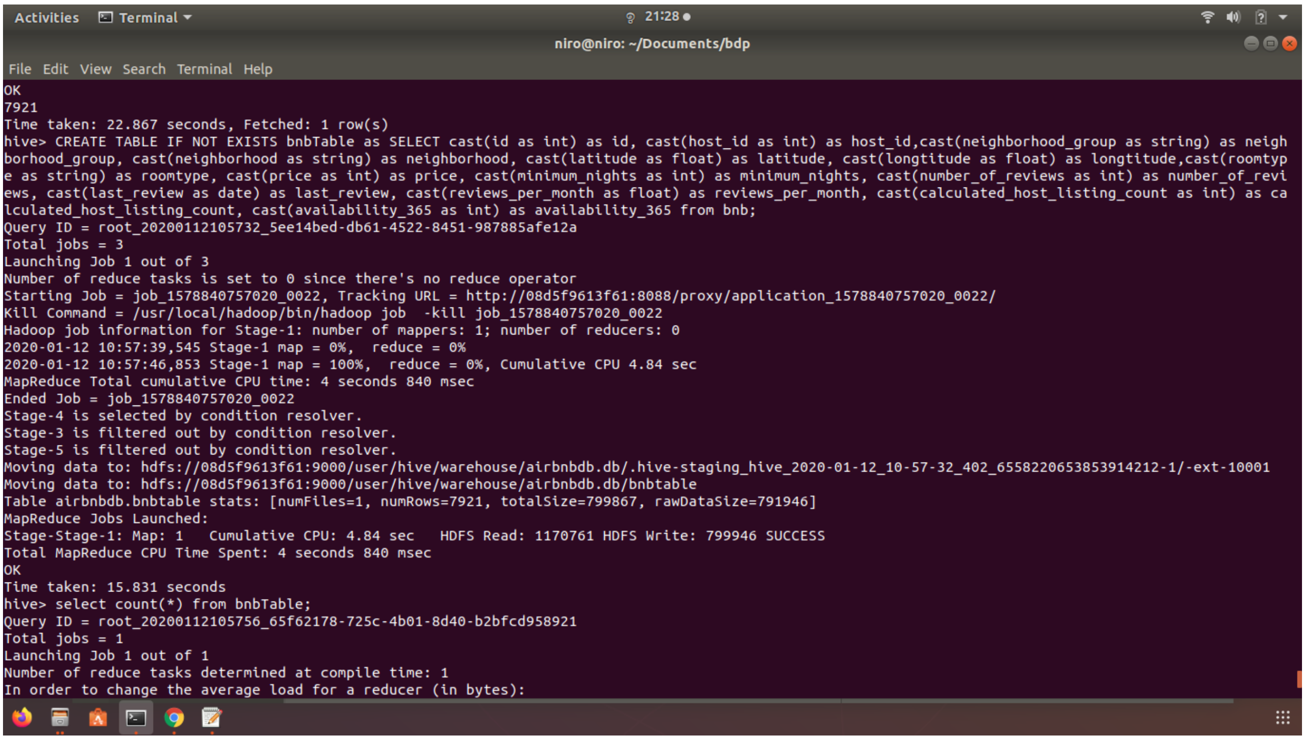
Activities
Terminal
21:27
niro@niro: ~/Documents/bdp
File Edit View Search Terminal Help

```
bash-4.1# hive

Logging initialized using configuration in jar:file:/usr/local/apache-hive-1.2.2-bin/lib/hive-common-1.2.2.jar!/hive-log4j.properties
hive> CREATE DATABASE IF NOT EXISTS airBnbDatabase;
OK
Time taken: 0.646 seconds
hive> USE airBnbDatabase;
OK
Time taken: 0.026 seconds
hive> CREATE EXTERNAL TABLE IF NOT EXISTS bnb(id INT, name STRING, host_id INT, host_name STRING, neighborhood_group STRING, neighborhood STRING, lati
tude FLOAT, longtitude FLOAT, roomtype STRING, price INT, minimum_nights INT, number_of_reviews INT, last_review DATE, reviews_per_month FLOAT, calcul
ated_host_listing_count INT, availability_365 INT)
    > ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
    > LOCATION '/airBnbData/'
    > TBLPROPERTIES ("skip.header.line.count"="1");
OK
Time taken: 0.392 seconds
hive> CREATE TABLE IF NOT EXISTS bnbTable as SELECT cast(id as int) as id, cast(host_id as int) as host_id,cast(neighborhood_group as string) as neigh
borhood_group, cast(neighborhood as string) as neighborhood, cast(latitude as float) as latitude, cast(longtitude as float) as longtitude,cast(roomtyp
e as string) as roomtype, cast(price as int) as price, cast(minimum_nights as int) as minimum_nights, cast(number_of_reviews as int) as number_of_revi
ews, cast(last_review as date) as last_review, cast(reviews_per_month as float) as reviews_per_month, cast(calculated_host_listing_count as int) as ca
lculated_host_listing_count, cast(availability_365 as int) as availability_365 from bnb;
OK
Time taken: 0.066 seconds
hive> select count(*) from bnbTable;
Query ID = root_20200112105500_73773c02-e471-4749-ad4e-ec677f6050fc
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1578840757020_0020, Tracking URL = http://08d5f9613f61:8088/proxy/application_1578840757020_0020/
```



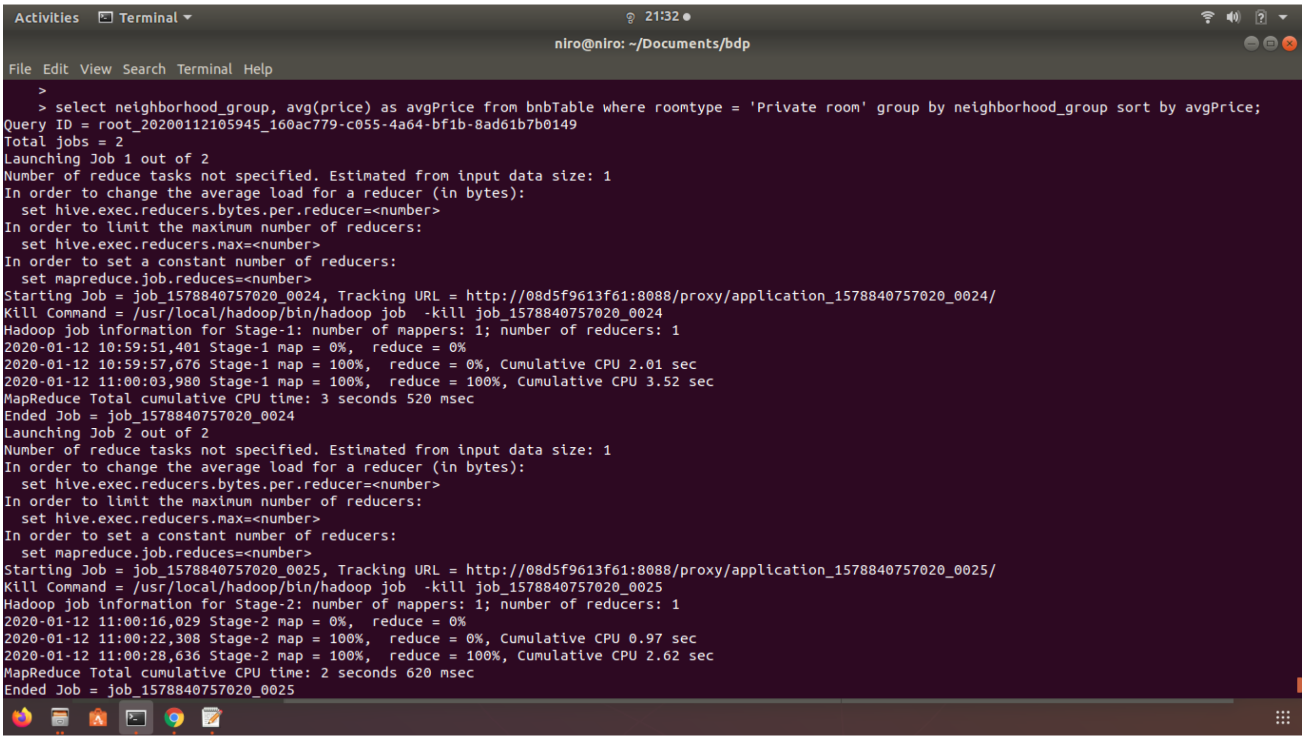
Activities
Terminal
21:27
niro@niro: ~/Documents/bdp
File Edit View Search Terminal Help

```
OK
Time taken: 0.081 seconds
hive> CREATE EXTERNAL TABLE IF NOT EXISTS bnb(id INT, name STRING, host_id INT, host_name STRING, neighborhood_group STRING, neighborhood STRING, lati
tude FLOAT, longtitude FLOAT, roomtype STRING, price INT, minimum_nights INT, number_of_reviews INT, last_review DATE, reviews_per_month FLOAT, calcul
ated_host_listing_count INT, availability_365 INT)
    > ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
    > LOCATION '/airBnbData/'
    > TBLPROPERTIES ("skip.header.line.count"="1");
OK
Time taken: 0.371 seconds
hive> select count(*) from bnb;
Query ID = root_20200112105646_34967b97-c910-47fd-b8ac-21a2056b2e3d
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1578840757020_0021, Tracking URL = http://08d5f9613f61:8088/proxy/application_1578840757020_0021/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1578840757020_0021
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-01-12 10:56:55,094 Stage-1 map = 0%,  reduce = 0%
2020-01-12 10:57:00,440 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.94 sec
2020-01-12 10:57:07,812 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.6 sec
MapReduce Total cumulative CPU time: 3 seconds 600 msec
Ended Job = job_1578840757020_0021
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.6 sec   HDFS Read: 1173277 HDFS Write: 5 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 600 msec
OK
7921
Time taken: 22.867 seconds, Fetched: 1 row(s)
hive> 
```


Activities
Terminal
21:28
niro@niro: ~/Documents/bdp
File Edit View Search Terminal Help

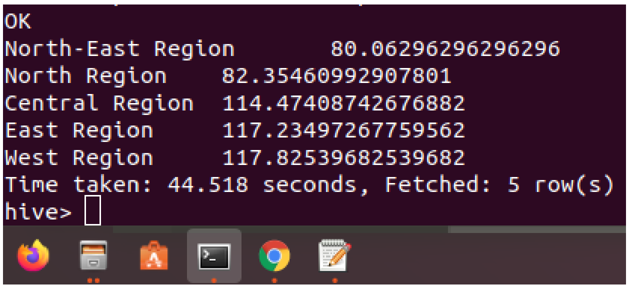
```
OK
7921
Time taken: 22.867 seconds, Fetched: 1 row(s)
hive> CREATE TABLE IF NOT EXISTS bnbTable as SELECT cast(id as int) as id, cast(host_id as int) as host_id,cast(neighborhood_group as string) as neigh
borhood_group, cast(neighborhood as string) as neighborhood, cast(latitude as float) as latitude, cast(longtitude as float) as longtitude,cast(roomtyp
e as string) as roomtype, cast(price as int) as price, cast(minimum_nights as int) as minimum_nights, cast(number_of_reviews as int) as number_of_revi
ews, cast(last_review as date) as last_review, cast(reviews_per_month as float) as reviews_per_month, cast(calculated_host_listing_count as int) as ca
lculated_host_listing_count, cast(availability_365 as int) as availability_365 from bnb;
Query ID = root_20200112105732_5ee14bed-db61-4522-8451-987885afe12a
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1578840757020_0022, Tracking URL = http://08d5f9613f61:8088/proxy/application_1578840757020_0022/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1578840757020_0022
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2020-01-12 10:57:39,545 Stage-1 map = 0%,  reduce = 0%
2020-01-12 10:57:46,853 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.84 sec
MapReduce Total cumulative CPU time: 4 seconds 840 msec
Ended Job = job_1578840757020_0022
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://08d5f9613f61:9000/user/hive/warehouse/airbnbdb.db/.hive-staging_hive_2020-01-12_10-57-32_402_6558220653853914212-1/-ext-10001
Moving data to: hdfs://08d5f9613f61:9000/user/hive/warehouse/airbnbdb.db/bnbtable
Table airbnbdb.bnbtable stats: [numFiles=1, numRows=7921, totalSize=799867, rawDataSize=791946]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 4.84 sec   HDFS Read: 1170761 HDFS Write: 799946 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 840 msec
OK
Time taken: 15.831 seconds
hive> select count(*) from bnbTable;
Query ID = root_20200112105756_65f62178-725c-4b01-8d40-b2bfcd958921
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
```

Question 01 - Average price of Private room rental by neighborhood group


Activities
Terminal
21:32
niro@niro: ~/Documents/bdp
File Edit View Search Terminal Help

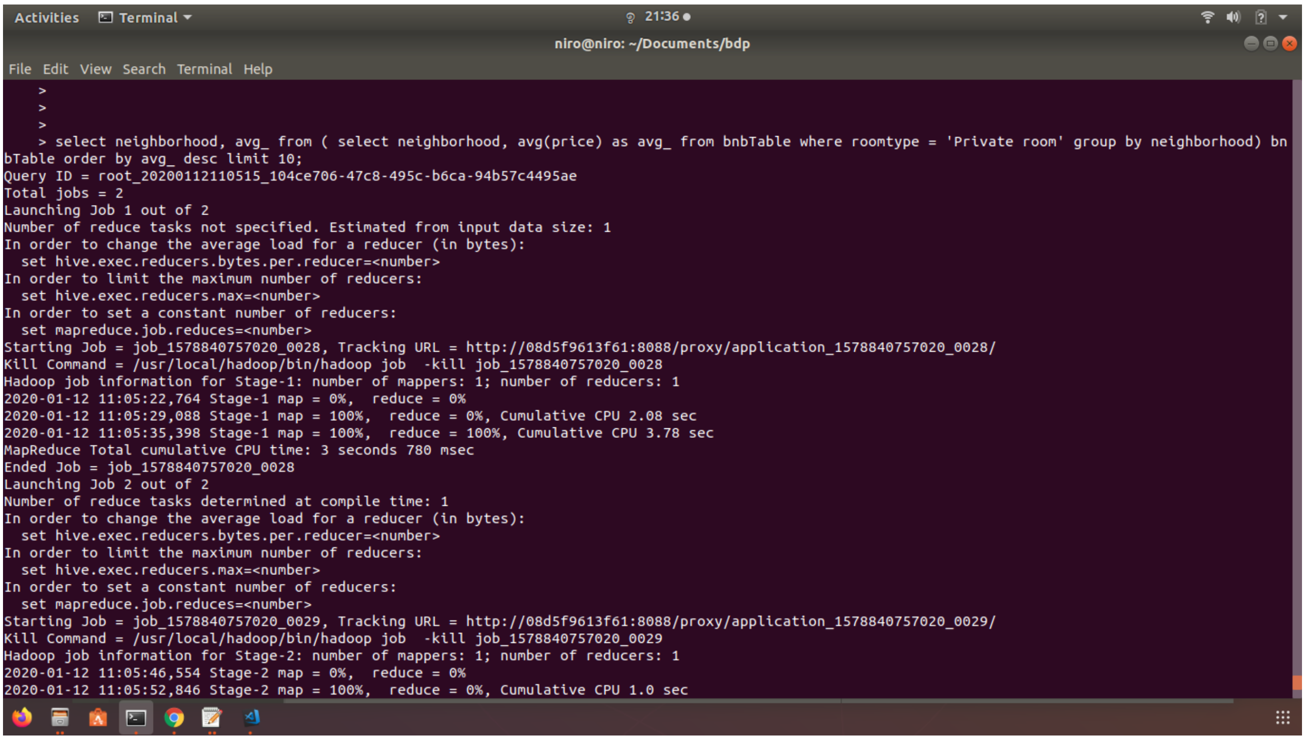
```
    >
    > select neighborhood_group, avg(price) as avgPrice from bnbTable where roomtype = 'Private room' group by neighborhood_group sort by avgPrice;
Query ID = root_20200112105945_160ac779-c055-4a64-bf1b-8ad61b7b0149
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1578840757020_0024, Tracking URL = http://08d5f9613f61:8088/proxy/application_1578840757020_0024/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1578840757020_0024
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-01-12 10:59:51,401 Stage-1 map = 0%,  reduce = 0%
2020-01-12 10:59:57,676 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.01 sec
2020-01-12 11:00:03,980 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.52 sec
MapReduce Total cumulative CPU time: 3 seconds 520 msec
Ended Job = job_1578840757020_0024
Launching Job 2 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1578840757020_0025, Tracking URL = http://08d5f9613f61:8088/proxy/application_1578840757020_0025/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1578840757020_0025
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2020-01-12 11:00:16,029 Stage-2 map = 0%,  reduce = 0%
2020-01-12 11:00:22,308 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 0.97 sec
2020-01-12 11:00:28,636 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 2.62 sec
MapReduce Total cumulative CPU time: 2 seconds 620 msec
Ended Job = job_1578840757020_0025
```

Question 01: output



```
OK
North-East Region       80.06296296296296
North Region    82.35460992907801
Central Region  114.47408742676882
East Region     117.23497267759562
West Region     117.82539682539682
Time taken: 44.518 seconds, Fetched: 5 row(s)
hive> 
```

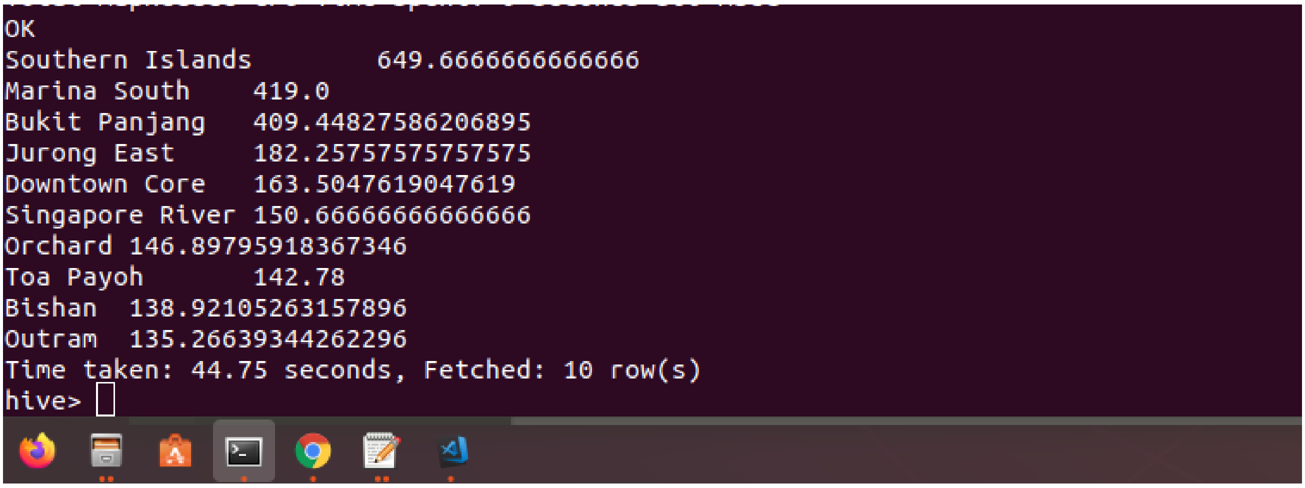
Question 02 - Top 10 neighborhood based on Average price of Private room



Activities
Terminal
21:36
niro@niro: ~/Documents/bdp
File Edit View Search Terminal Help

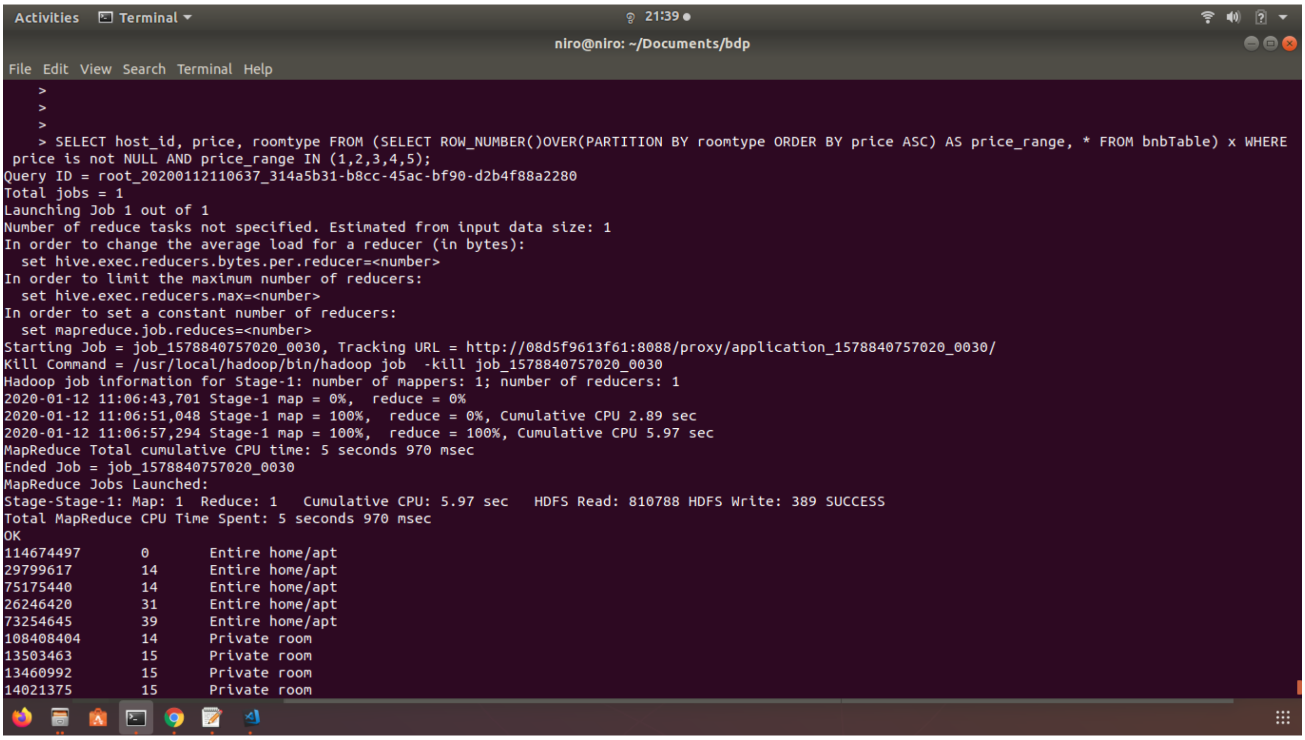
```
    >
    >
    >
    > select neighborhood, avg_ from ( select neighborhood, avg(price) as avg_ from bnbTable where roomtype = 'Private room' group by neighborhood) bn
bTable order by avg_ desc limit 10;
Query ID = root_20200112110515_104ce706-47c8-495c-b6ca-94b57c4495ae
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1578840757020_0028, Tracking URL = http://08d5f9613f61:8088/proxy/application_1578840757020_0028/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1578840757020_0028
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-01-12 11:05:22,764 Stage-1 map = 0%,  reduce = 0%
2020-01-12 11:05:29,088 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.08 sec
2020-01-12 11:05:35,398 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.78 sec
MapReduce Total cumulative CPU time: 3 seconds 780 msec
Ended Job = job_1578840757020_0028
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1578840757020_0029, Tracking URL = http://08d5f9613f61:8088/proxy/application_1578840757020_0029/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1578840757020_0029
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2020-01-12 11:05:46,554 Stage-2 map = 0%,  reduce = 0%
2020-01-12 11:05:52,846 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 1.0 sec
```

Question 02: output



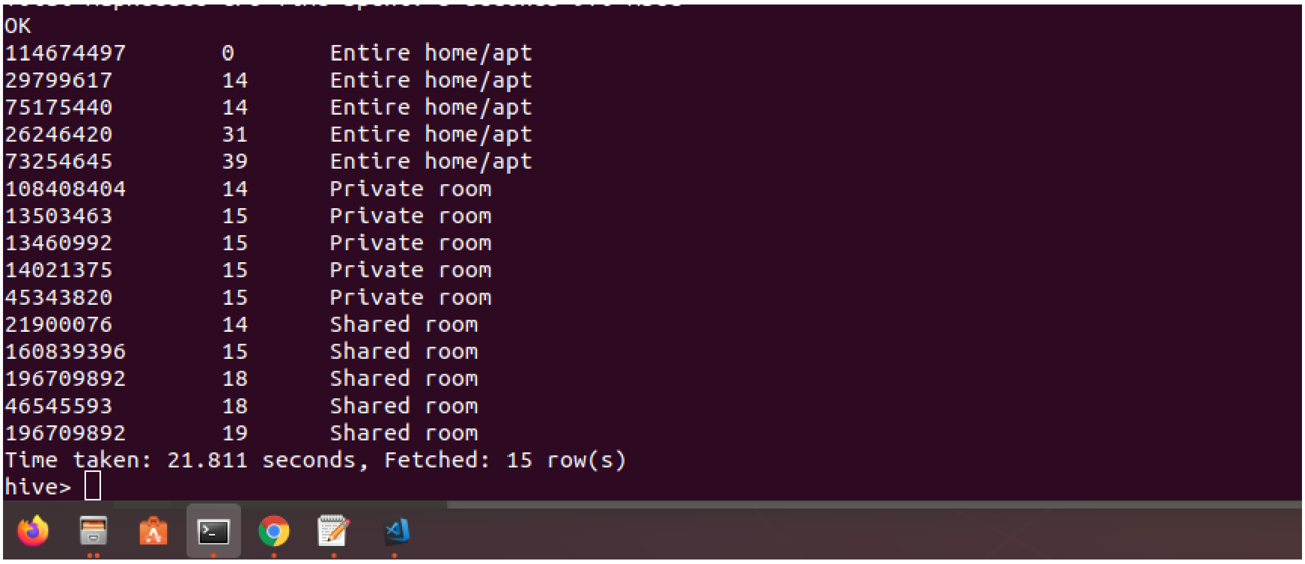
```
OK
Southern Islands        649.6666666666666
Marina South    419.0
Bukit Panjang   409.44827586206895
Jurong East     182.25757575757575
Downtown Core   163.5047619047619
Singapore River 150.66666666666666
Orchard 146.89795918367346
Toa Payoh       142.78
Bishan  138.92105263157896
Outram  135.26639344262296
Time taken: 44.75 seconds, Fetched: 10 row(s)
hive> 
```

Question 3 - The 5 lowest price properties per each Room Type


Activities
Terminal
21:39
niro@niro: ~/Documents/bdp
File Edit View Search Terminal Help

```
    >
    >
    >
    > SELECT host_id, price, roomtype FROM (SELECT ROW_NUMBER()OVER(PARTITION BY roomtype ORDER BY price ASC) AS price_range, * FROM bnbTable) x WHERE
 price is not NULL AND price_range IN (1,2,3,4,5);
Query ID = root_20200112110637_314a5b31-b8cc-45ac-bf90-d2b4f88a2280
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1578840757020_0030, Tracking URL = http://08d5f9613f61:8088/proxy/application_1578840757020_0030/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1578840757020_0030
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-01-12 11:06:43,701 Stage-1 map = 0%,  reduce = 0%
2020-01-12 11:06:51,048 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.89 sec
2020-01-12 11:06:57,294 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 5.97 sec
MapReduce Total cumulative CPU time: 5 seconds 970 msec
Ended Job = job_1578840757020_0030
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.97 sec   HDFS Read: 810788 HDFS Write: 389 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 970 msec
OK
114674497       0       Entire home/apt
29799617        14      Entire home/apt
75175440        14      Entire home/apt
26246420        31      Entire home/apt
73254645        39      Entire home/apt
108408404       14      Private room
13503463        15      Private room
13460992        15      Private room
14021375        15      Private room
```

Question 03: output



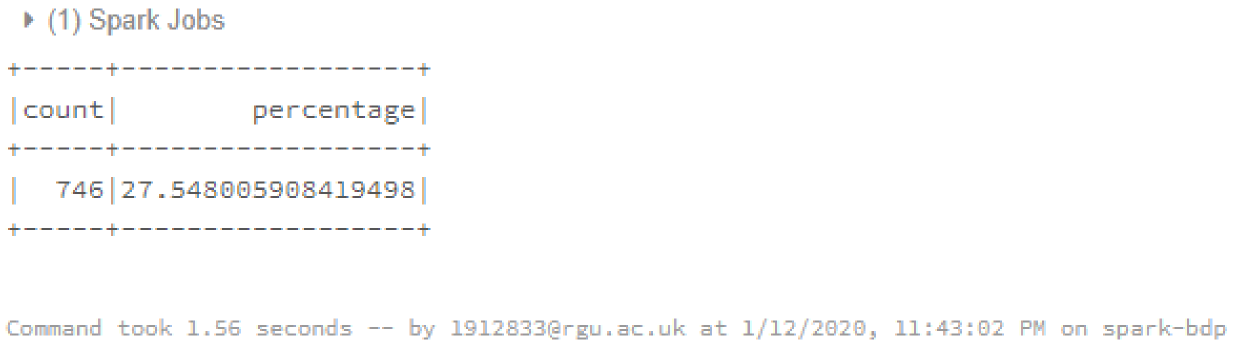
```
OK
114674497       0       Entire home/apt
29799617        14      Entire home/apt
75175440        14      Entire home/apt
26246420        31      Entire home/apt
73254645        39      Entire home/apt
108408404       14      Private room
13503463        15      Private room
13460992        15      Private room
14021375        15      Private room
45343820        15      Private room
21900076        14      Shared room
160839396       15      Shared room
196709892       18      Shared room
46545593        18      Shared room
196709892       19      Shared room
Time taken: 21.811 seconds, Fetched: 15 row(s)
hive> 
```

2.2. Spark

Question 1 - Percentage of owners who rent more than one property

```
// insert the data from csv
import org.apache.spark.sql.functions.{col, to_date}
import org.apache.spark.sql.functions._

var upDF=spark.read
          .option("header", "true")
          .option("treatEmptyValuesAsNulls", "true")
          .option("mode","DROPMALFORMED")
          .option("delimiter", ",")
          .option("inferSchema", "true")
          .csv("/FileStore/tables/listings.csv")

//convert the string format to date in date columns
val df = upDF.columns.filter(colName =>colName.endsWith("_review"))
.foldLeft(upDF) { (outputDF, columnName) =>
outputDF.withColumn(columnName, to_date(col(columnName), "MM/dd/yyyy").cast("date"))
}

//write data into data frame
var rentalDf = df.toDF();

//drop null values
val totalHostsIds = rentalDf.filter(rentalDf("host_id").isNotNull).select("host_id")
.distinct().count()

//hosts that contains more than one rentals
rentalDf = rentalDf.where("calculated_host_listings_count >1").select("host_id").dis
tinct()

//Count the number of hosts that contains more than one rentals
rentalDf = rentalDf.groupBy("host_id").count().agg(count("host_id").alias("count"))

// Convert the hosts numbers by 100 to produce percentage
val udf_host_percentage = udf((x:Int)=>{(x*100)/totalHostsIds.toDouble})

//show results by adding new value named percentage
rentalDf.withColumn("percentage",udf_host_percentage(rentalDf("count"))).show()
```

Question 01: output



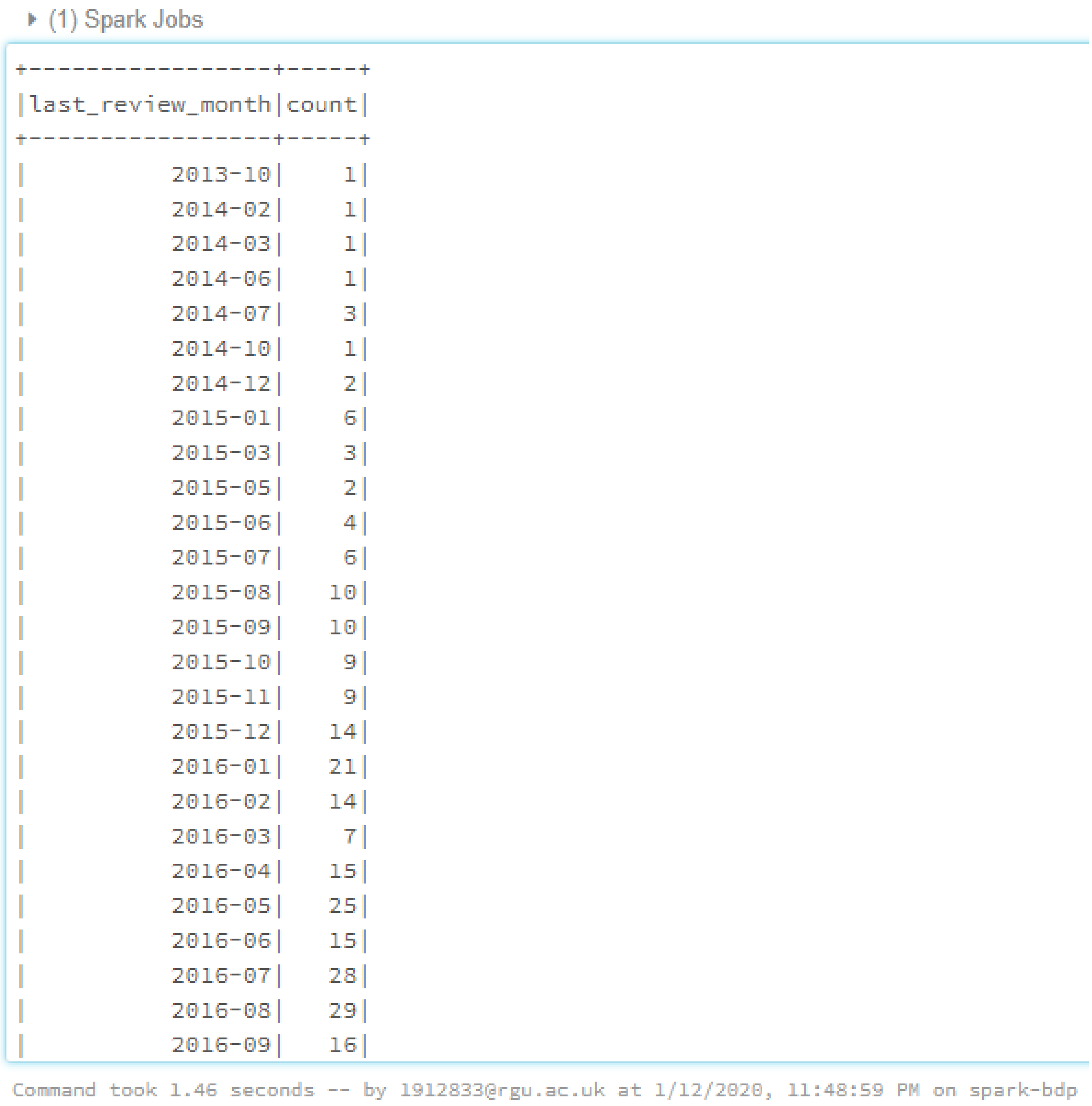
```
▸ (1) Spark Jobs
+-----+------------------+
|count|        percentage|
+-----+------------------+
|  746|27.548005908419498|
+-----+------------------+


Command took 1.56 seconds -- by 1912833@rgu.ac.uk at 1/12/2020, 11:43:02 PM on spark-bdp
```

Question 02: Histogram of number of rentals reviewed over time (based on last review) in mouth granularity.

```
// insert the data from csv
import org.apache.spark.sql.functions.{col, to_date}

var upDF=spark.read
          .option("header", "true")
          .option("treatEmptyValuesAsNulls", "true")
          .option("mode","DROPMALFORMED")
          .option("delimiter", ",")
          .option("inferSchema", "true")
          .csv("/FileStore/tables/listings.csv")

// convert the string format to date in date columns
val df = upDF.columns.filter(colName =>colName.endsWith("_review"))
.foldLeft(upDF) { (outputDF, columnName) =>
outputDF.withColumn(columnName, to_date(col(columnName), "MM/dd/yyyy").cast("date"))
}

df.count

//write data into data frame
var rentalDf = df.toDF();

//drop null values
rentalDf = rentalDf.select("last_review").where("last_review IS NOT NULL")

//get year with month substring "yyyy-MM"
val udf_get_month = udf((x:String)=>x.slice(0,7))

//add new column called last_review_month
```

```
rentalDf = rentalDf.withColumn("last_review_month",udf_get_month(rentalDf("last_review")))

//group last review
//display count per month granularity
rentalDf.groupBy("last_review_month").count().alias("review_count").sort("last_review_month").count

//group last review and show count per month granularity
rentalDf.groupBy("last_review_month").count().alias("review_count").sort("last_review_month").show(61)
```

Question 02: output



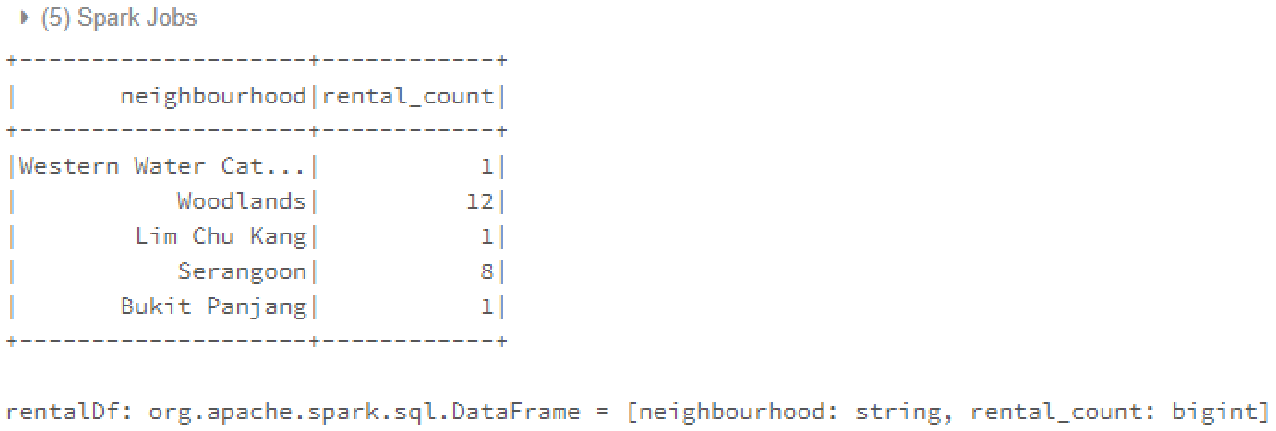
```
+-----------------+-----+
|last_review_month|count|
+-----------------+-----+
|          2013-10|    1|
|          2014-02|    1|
|          2014-03|    1|
|          2014-06|    1|
|          2014-07|    3|
|          2014-10|    1|
|          2014-12|    2|
|          2015-01|    6|
|          2015-03|    3|
|          2015-05|    2|
|          2015-06|    4|
|          2015-07|    6|
|          2015-08|   10|
|          2015-09|   10|
|          2015-10|    9|
|          2015-11|    9|
|          2015-12|   14|
|          2016-01|   21|
|          2016-02|   14|
|          2016-03|    7|
|          2016-04|   15|
|          2016-05|   25|
|          2016-06|   15|
|          2016-07|   28|
|          2016-08|   29|
|          2016-09|   16|
```

Command took 1.46 seconds -- by 1912833@rgu.ac.uk at 1/12/2020, 11:48:59 PM on spark-bdp

Question 03 - Number of rentals that are available all 365 days of the year for each neighborhood, that are in the neighborhood which have top 5 average rental prices.

```
// insert the data from csv
import org.apache.spark.sql.functions.{col, to_date}

var upDF=spark.read
          .option("header", "true")
          .option("treatEmptyValuesAsNulls", "true")
          .option("mode","DROPMALFORMED")
          .option("delimiter", ",")
          .option("inferSchema", "true")
          .csv("/FileStore/tables/listings.csv")

//convert the string format to date in date columns
val df = upDF.columns.filter(colName =>colName.endsWith("_review"))
.foldLeft(upDF) { (outputDF, columnName) =>
outputDF.withColumn(columnName, to_date(col(columnName), "MM/dd/yyyy").cast("date"))
}

//write data into data frame
var rentalDf = df.toDF();

//filter dataframe that availability equals 365
rentalDf = rentalDf.where("availability_365 = 365");

//create temporary view to store data
rentalDf.createOrReplaceTempView("Available356DaysView")

rentalDf=rentalDf.sqlContext.sql("SELECT neighbourhood, COUNT(*) AS rental_count FRO
M Available356DaysView" +
          " WHERE neighbourhood IN (SELECT neighbourhood FROM Available356DaysView G
ROUP BY neighbourhood ORDER BY avg(price)" +
          " LIMIT 5) GROUP BY neighbourhood")

rentalDf.show(10)
```

Question 03: output



```
+--------------------+------------+
|       neighbourhood|rental_count|
+--------------------+------------+
|Western Water Cat...|           1|
|            Woodlands|         12|
|         Lim Chu Kang|          1|
|            Serangoon|          8|
|        Bukit Panjang|          1|
+--------------------+------------+

rentalDf: org.apache.spark.sql.DataFrame = [neighbourhood: string, rental_count: bigint]
```

3. Steps for training and validating the model

01. Exploring The Data

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('ml-bank').getOrCreate()
df = spark.read.csv('FileStore/tables/listings.csv', header = True, inferSchema
= True)
df.printSchema()

totalCount = df.count()
```

```
root
 |-- id: string (nullable = true)
 |-- name: string (nullable = true)
 |-- host_id: string (nullable = true)
 |-- host_name: string (nullable = true)
 |-- neighbourhood_group: string (nullable = true)
 |-- neighbourhood: string (nullable = true)
 |-- latitude: double (nullable = true)
 |-- longitude: string (nullable = true)
 |-- room_type: string (nullable = true)
 |-- price: integer (nullable = true)
 |-- minimum_nights: integer (nullable = true)
 |-- number_of_reviews: string (nullable = true)
 |-- last_review: string (nullable = true)
 |-- reviews_per_month: double (nullable = true)
 |-- calculated_host_listings_count: integer (nullable = true)
 |-- availability_365: integer (nullable = true)
```

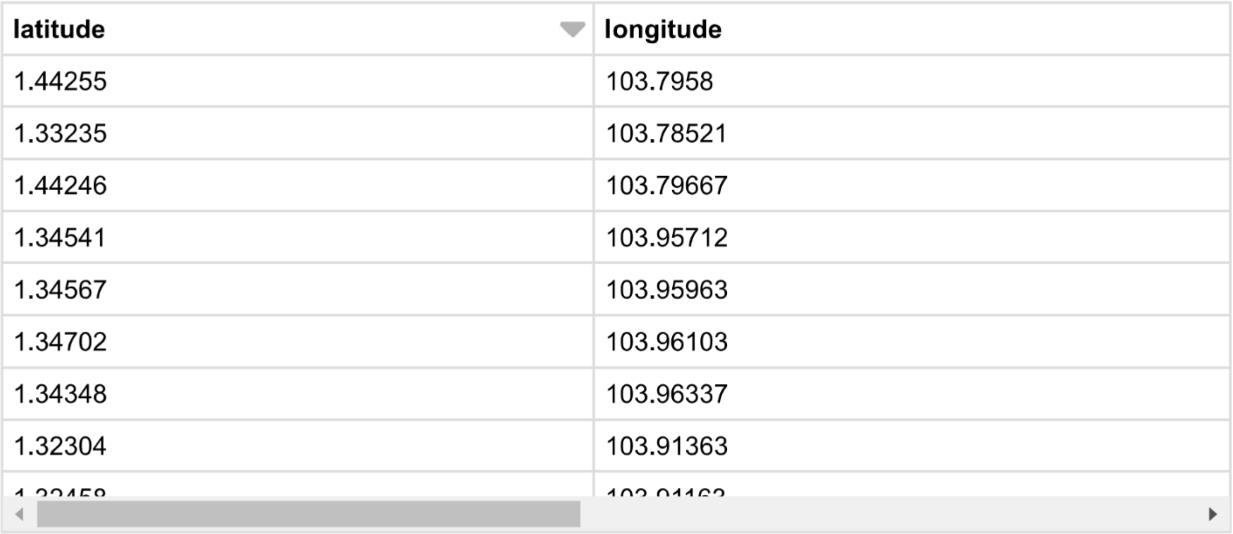
Input variables: Lat, Long **Output variable:** Neigbourhood group

```
# select input variables and output variables only
df = df.select('latitude','longitude', 'neighbourhood_group')
cols = df.columns
df.printSchema()

root
 |-- latitude: double (nullable = true)
 |-- longitude: string (nullable = true)
 |-- neighbourhood_group: string (nullable = true)

# convert str to double
df =
df.withColumn('latitude',df['latitude'].cast("double")).withColumn('longitude',
df['longitude'].cast("double"))
```

```
#drop null values
df = df.dropna()
nullValuesCount = totalCount - df.count()
nullValuesCount
```

```
Out[4]: 26
```

```
display(df)
```



latitude	longitude
1.44255	103.7958
1.33235	103.78521
1.44246	103.79667
1.34541	103.95712
1.34567	103.95963
1.34702	103.96103
1.34348	103.96337
1.32304	103.91363
1.32458	103.91163

Showing the first 1000 rows.

02. Preparing Data for Machine Learning

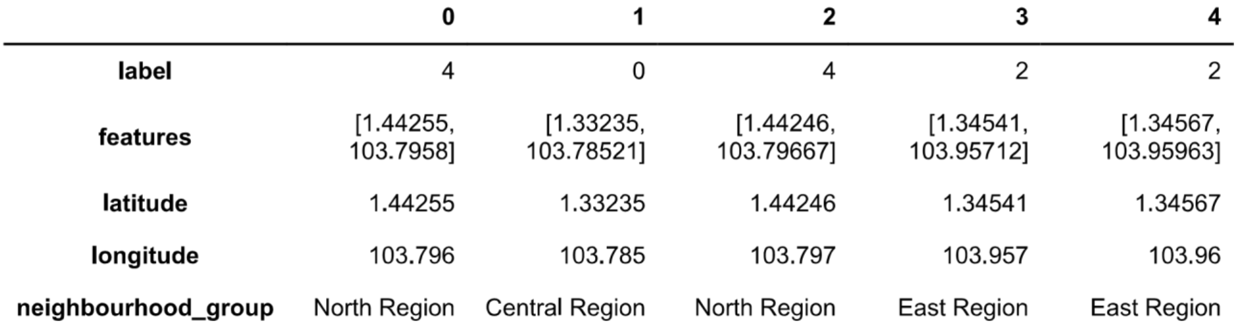
```
from pyspark.ml.feature import OneHotEncoderEstimator, StringIndexer,
VectorAssembler

stages = []

label_stringIdx = StringIndexer(inputCol = 'neighbourhood_group', outputCol =
'label')
stages += [label_stringIdx]

assemblerInputs = ['latitude', 'longitude']
assembler = VectorAssembler(inputCols=assemblerInputs, outputCol="features")
stages += [assembler]
```

```
# Pipeline
from pyspark.ml import Pipeline

pipeline = Pipeline(stages = stages)
pipelineModel = pipeline.fit(df)
df = pipelineModel.transform(df)
selectedCols = ['label', 'features'] + cols
df = df.select(selectedCols)
df.printSchema()
```

```
root
 |-- label: double (nullable = false)
 |-- features: vector (nullable = true)
 |-- latitude: double (nullable = true)
 |-- longitude: double (nullable = true)
 |-- neighbourhood_group: string (nullable = true)
```

```
import pandas as pd
pd.DataFrame(df.take(5), columns=df.columns).transpose()
```

```
Out[9]:
```



	0	1	2	3	4
label	4	0	4	2	2
features	[1.44255, 103.7958]	[1.33235, 103.78521]	[1.44246, 103.79667]	[1.34541, 103.95712]	[1.34567, 103.95963]
latitude	1.44255	1.33235	1.44246	1.34541	1.34567
longitude	103.796	103.785	103.797	103.957	103.96
neighbourhood_group	North Region	Central Region	North Region	East Region	East Region

```
# split data for testing and training

train, test = df.randomSplit([0.8, 0.2], seed = 2018)
print("Training Dataset Count: " + str(train.count()))
print("Test Dataset Count: " + str(test.count()))
```

```
Training Dataset Count: 6310
Test Dataset Count: 1585
```

03. Use the Logistic Regression Model

```
from pyspark.ml.classification import LogisticRegression

# We can also use the multinomial family for binary classification
mlr = LogisticRegression(maxIter=10, regParam=0.3, elasticNetParam=0.8,
family="multinomial")

# Fit the model
mlrModel = mlr.fit(train)

# Print the coefficients and intercepts for logistic regression with
multinomial family
print("Multinomial coefficients: " + str(mlrModel.coefficientMatrix))
print("Multinomial intercepts: " + str(mlrModel.interceptVector))
```

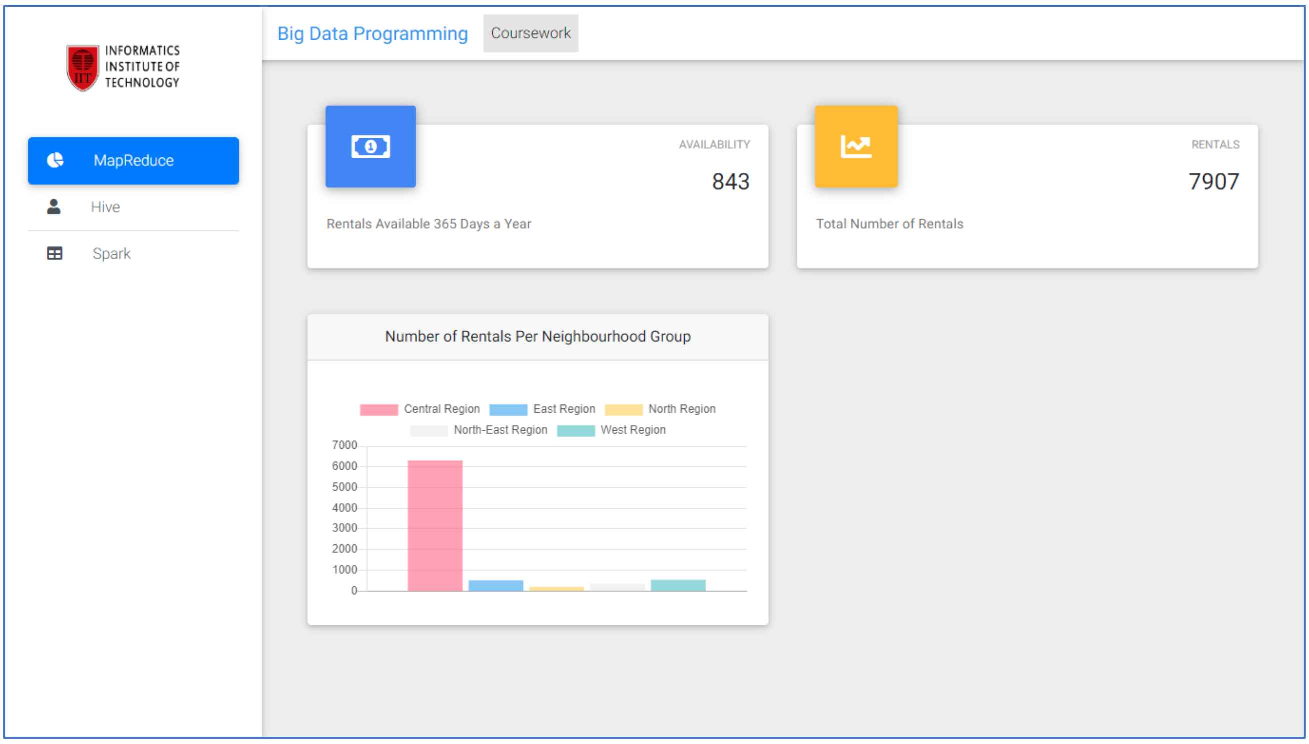
```
Multinomial coefficients: 5 X 2 CSCMatrix
(0,0) -0.0787
(0,1) 0.0009
Multinomial intercepts: [2.2726216535772634,-0.21051363042399887,-0.26405435986
55797,-0.6423941083280993,-1.1556595549595854]
```

```
# Make predictions on the test set

predictions = mlrModel.transform(test)
predictions.select('latitude', 'longitude', 'label', 'rawPrediction',
'prediction', 'probability').show(10)
```

```
+--------+---------+-----+--------------------+----------+--------------------+
|latitude|longitude|label|       rawPrediction|prediction|         probability|
+--------+---------+-----+--------------------+----------+--------------------+
| 1.25284|103.82225|  0.0|[2.27224689422807...|       0.0|[0.80041664889155...|
| 1.26624|103.81097|  0.0|[2.27118185140204...|       0.0|[0.80024645403475...|
| 1.26675|103.81219|  0.0|[2.27114287625884...|       0.0|[0.80024022370464...|
| 1.26814|103.81203|  0.0|[2.27103335336930...|       0.0|[0.80022271525892...|
| 1.26863| 103.8239|  0.0|[2.27100602564435...|       0.0|[0.80021834644022...|
| 1.26983|103.81331|  0.0|[2.27090158719125...|       0.0|[0.80020164945145...|
| 1.27173|103.82232|  0.0|[2.27076060898284...|       0.0|[0.80017910904683...|
| 1.27234|103.83224|  0.0|[2.27072199462472...|       0.0|[0.80017293482891...|
| 1.27237|103.83233|  0.0|[2.27071971921782...|       0.0|[0.80017257099973...|
| 1.27239|103.83419|  0.0|[2.27071990488162...|       0.0|[0.80017260068670...|
+--------+---------+-----+--------------------+----------+--------------------+
only showing top 10 rows
```

```
#Evaluate our Logistic Regression model.
from pyspark.ml.evaluation import BinaryClassificationEvaluator

evaluator = BinaryClassificationEvaluator()
print('Test Area Under ROC', evaluator.evaluate(predictions))

Test Area Under ROC 0.5
```

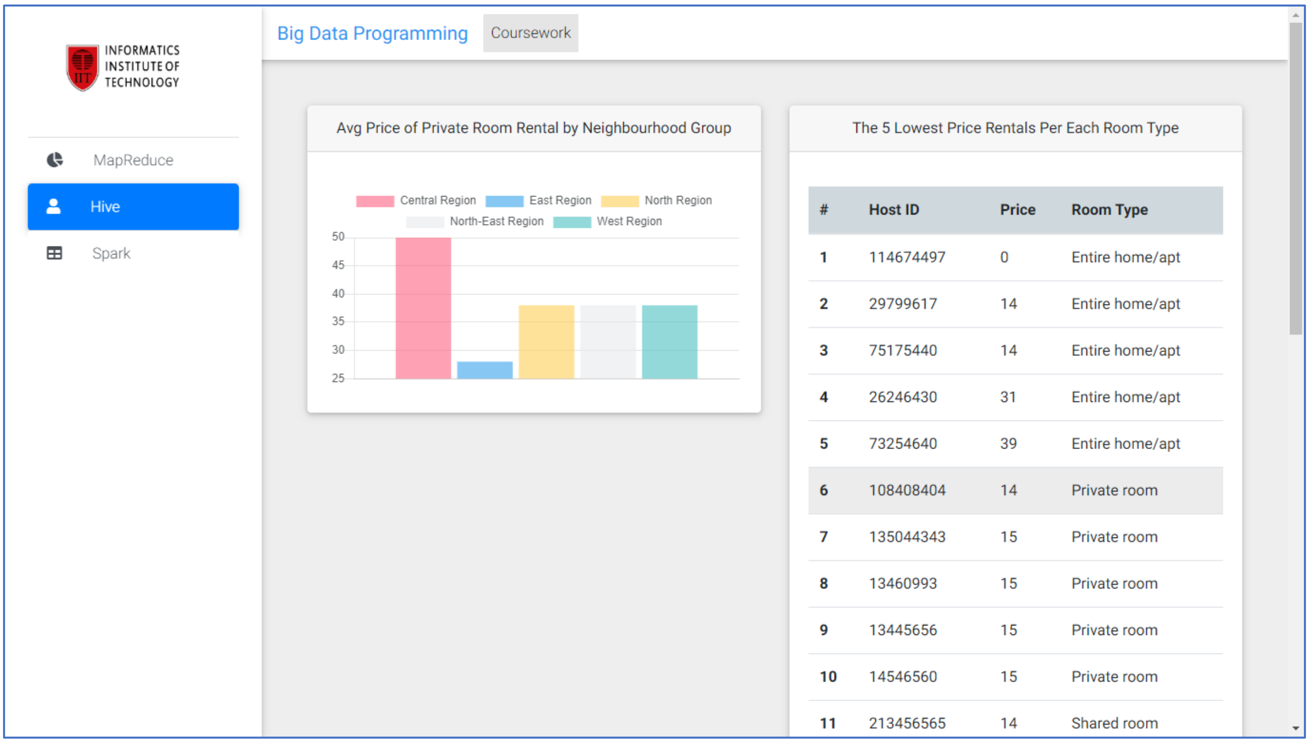
4. Screenshots of the Dashboard



AVAILABILITY
843
Rentals Available 365 Days a Year

RENTALS
7907
Total Number of Rentals

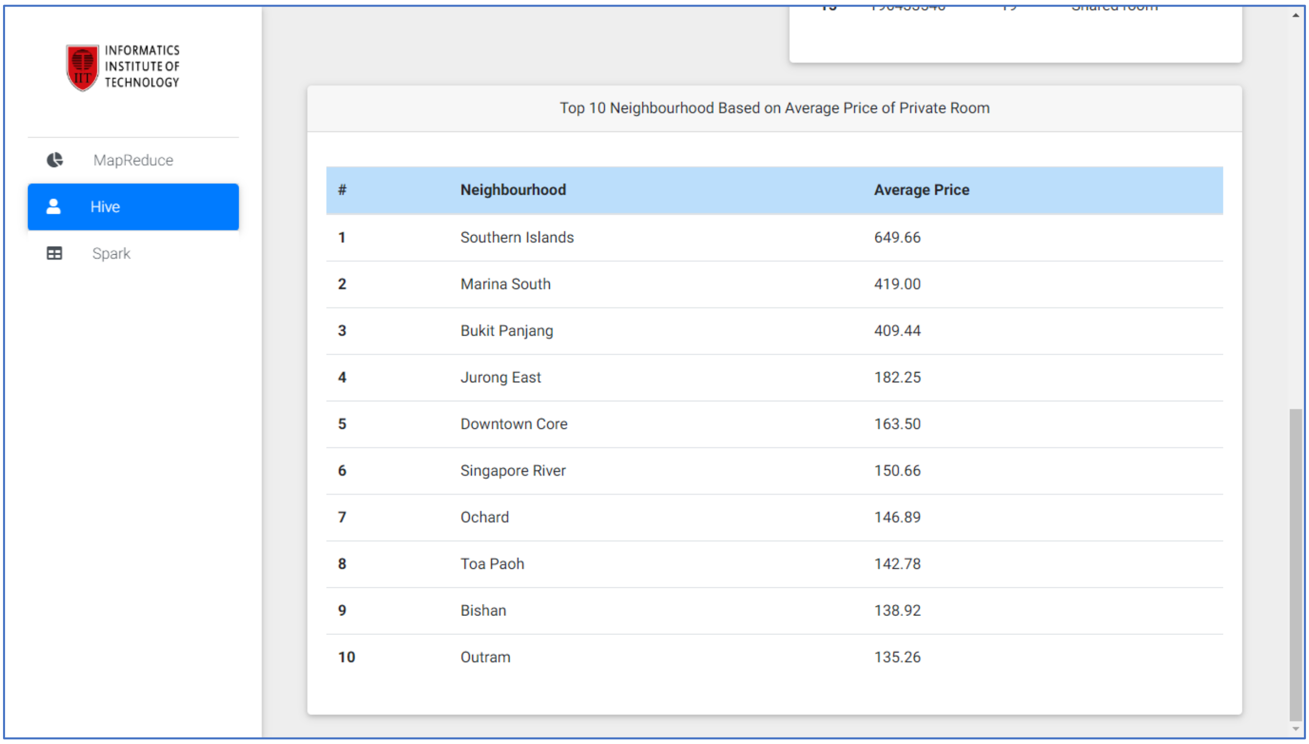
Number of Rentals Per Neighbourhood Group



Avg Price of Private Room Rental by Neighbourhood Group

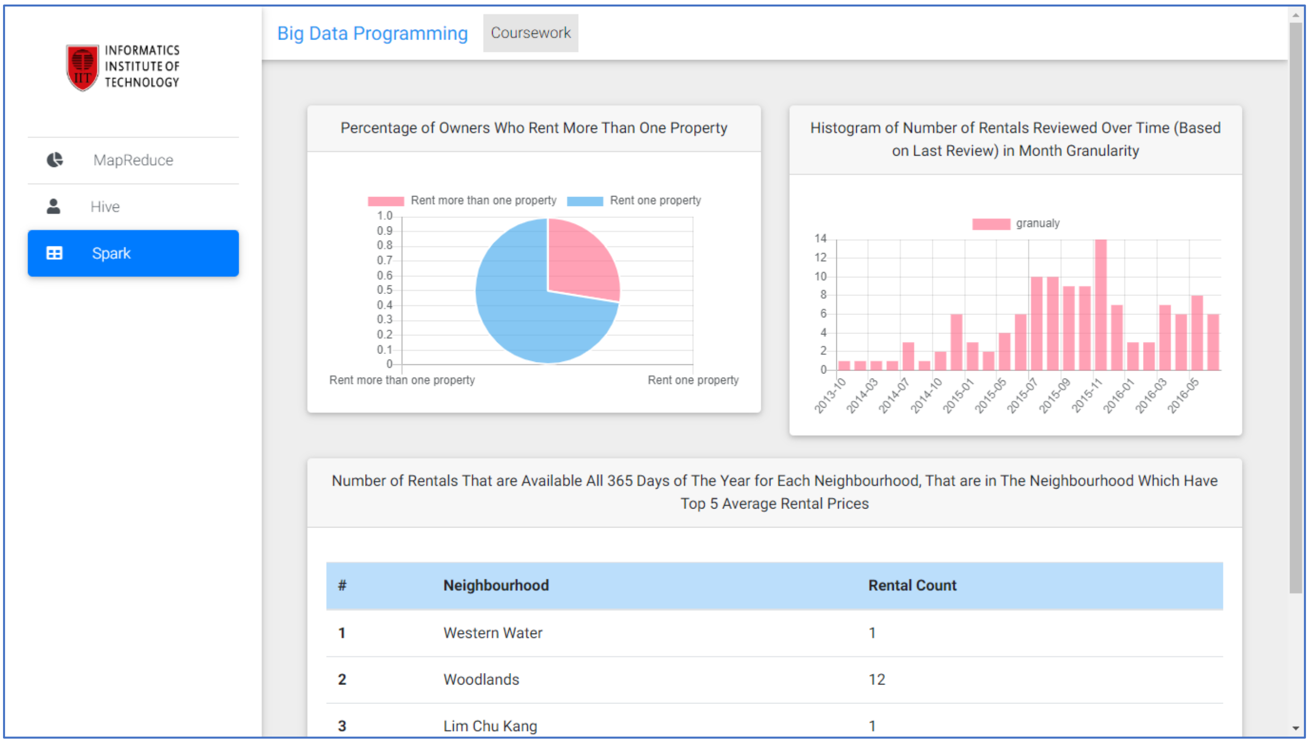
The 5 Lowest Price Rentals Per Each Room Type

#	Host ID	Price	Room Type
1	114674497	0	Entire home/apt
2	29799617	14	Entire home/apt
3	75175440	14	Entire home/apt
4	26246430	31	Entire home/apt
5	73254640	39	Entire home/apt
6	108408404	14	Private room
7	135044343	15	Private room
8	13460993	15	Private room
9	13445656	15	Private room
10	14546560	15	Private room
11	213456565	14	Shared room



Top 10 Neighbourhood Based on Average Price of Private Room

#	Neighbourhood	Average Price
1	Southern Islands	649.66
2	Marina South	419.00
3	Bukit Panjang	409.44
4	Jurong East	182.25
5	Downtown Core	163.50
6	Singapore River	150.66
7	Ochard	146.89
8	Toa Paoh	142.78
9	Bishan	138.92
10	Outram	135.26

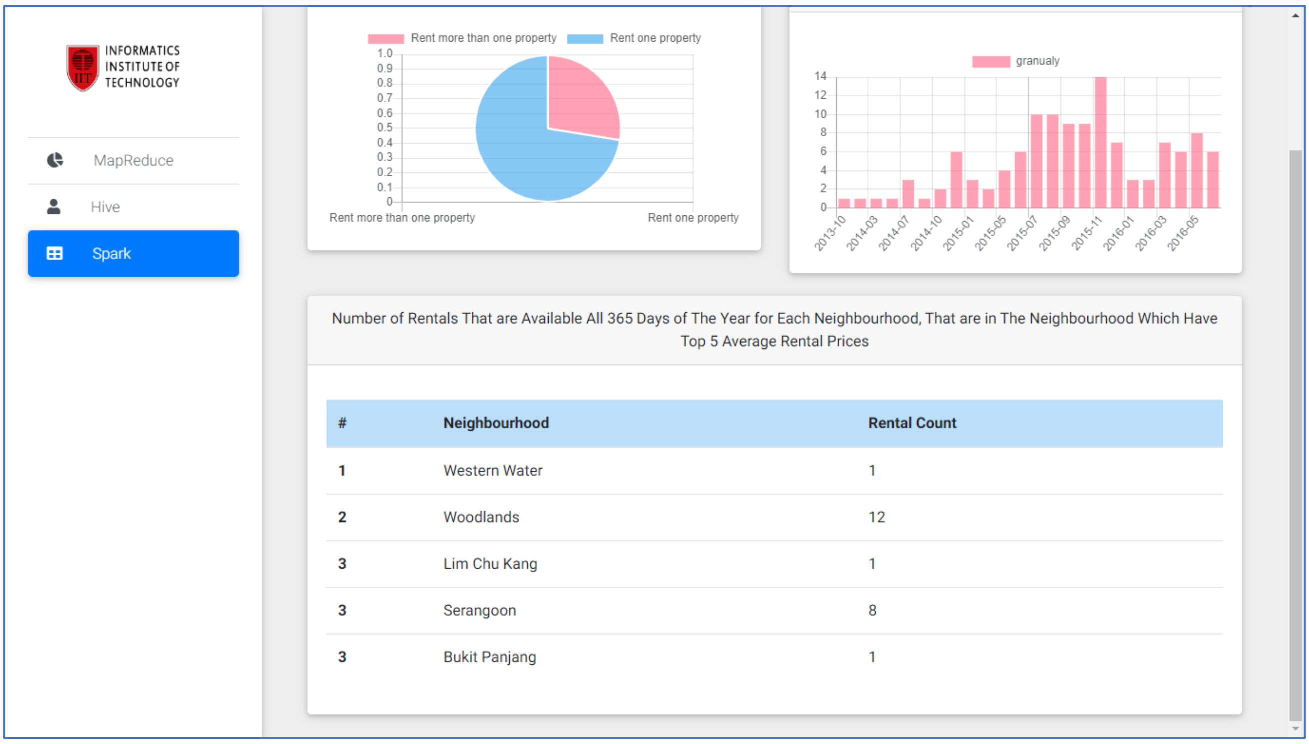


Percentage of Owners Who Rent More Than One Property

Histogram of Number of Rentals Reviewed Over Time (Based on Last Review) in Month Granularity

Number of Rentals That are Available All 365 Days of The Year for Each Neighbourhood, That are in The Neighbourhood Which Have Top 5 Average Rental Prices

#	Neighbourhood	Rental Count
1	Western Water	1
2	Woodlands	12
3	Lim Chu Kang	1



Number of Rentals That are Available All 365 Days of The Year for Each Neighbourhood, That are in The Neighbourhood Which Have Top 5 Average Rental Prices

#	Neighbourhood	Rental Count
1	Western Water	1
2	Woodlands	12
3	Lim Chu Kang	1
3	Serangoon	8
3	Bukit Panjang	1

Github Repository: https://github.com/niroshank/big-data-hive-mapred-spark