INFORMATICS INSTITUTE OF TECHNOLOGY

In Collaboration With

ROBERT GORDON UNIVERSITY ABERDEEN

MSc. in Big Data Analytics

2019/2020

By

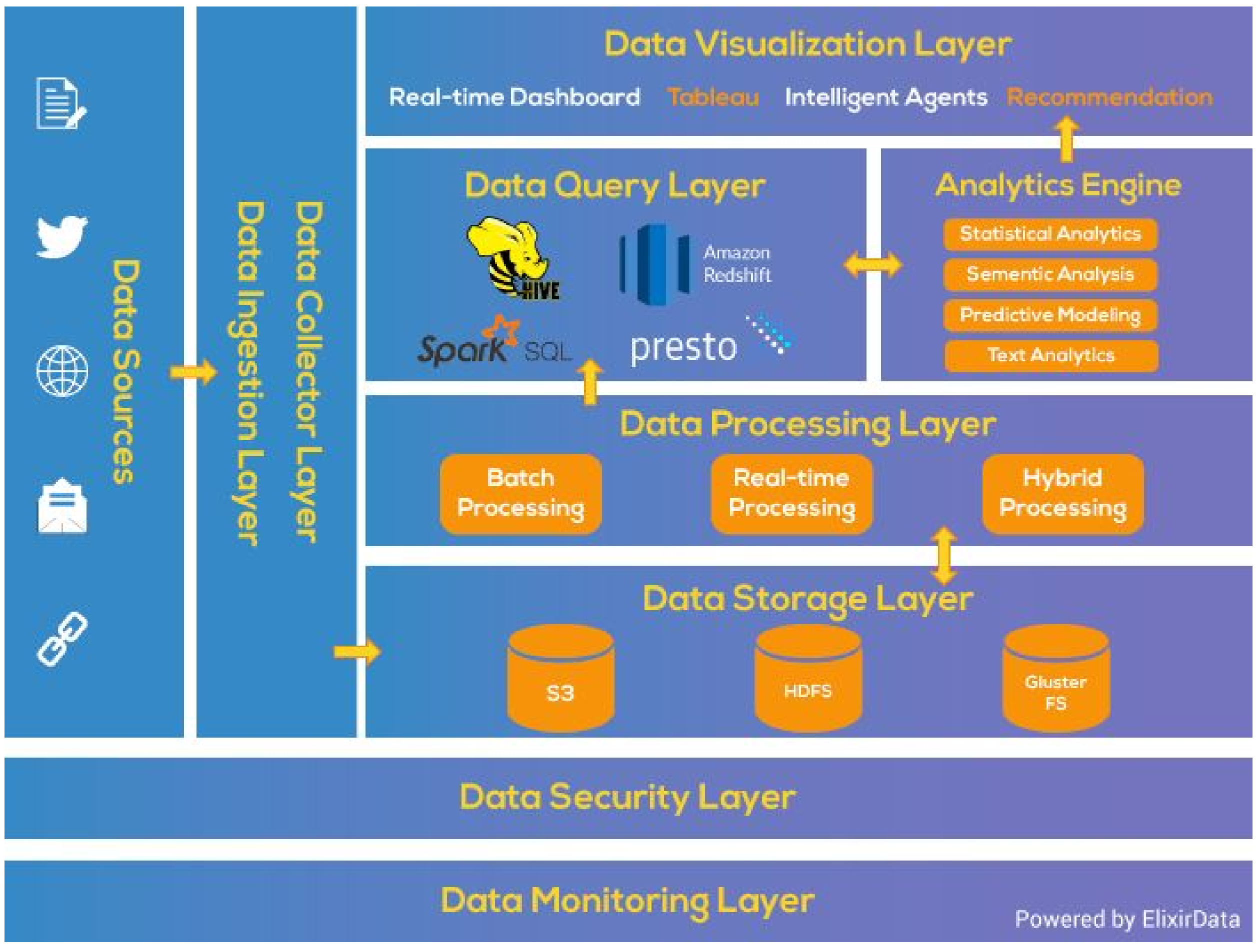
Laththuwa Handi Sanjaya De Silva

IIT No : **2019004** RGU Id : **1815301**

CMM705

Web Dashboard, Scripts, and other files are in the following GitHub Repository:
https://github.com/SanjayaDeSilva/bdp_coursework_2019004

Q1)Deployment Diagram



When we consider Big Data, normally it will be either Batch or Stream Processing Systems. Above architecture has 6 layers to ensure the flow of data.

1. **Data Ingestion Layer.**
   This is the 1st step where data coming from various sources for the journey.
2. **Data collector Layer.**
   This layer is where components are decoupled and start the pipeline for analytics.
3. **Data Storage Layer.**
   Where data store in various formats effectively for the processing.
4. **Data Processing Layer.**
   Real Time, Hybrid, Batch Processing happens in this layer.
5. **Data Query Layer.**
   Here happens analytics processing(spark, hive, ...). Focus here is gathering data in order To get help from them in the upcoming years.

6. **Data Visualization Layer.**
   This is where present the analyzed data to the end user using dashboards or other methods.

Q2) Analysis for Singapore Airbnb

Given data-set has imported to the data-lake as an external table. The Following screens show the formation of the external table in data-lake.

```
sanjaya@sanjaya:~/Dev/MSc/BDP$ sudo docker run -p 8088:8088 -p 50070:50070 -v /home/sanjaya/Dev/MSc/BDP:/resources --name hadoop-hive-pig -d suhothayan/hadoop-hive-pig:2.7.1
42ed2e4b750c0d16a2ebb5de10048fd24011b6bd2992c165a03964ae040d150f
sanjaya@sanjaya:~/Dev/MSc/BDP$ sudo docker exec -it hadoop-hive-pig bash
bash-4.1# hdfs dfs -mkdir /externaltablenew
20/01/10 01:56:09 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
mkdir: Cannot create directory /externaltablenew. Name node is in safe mode.
bash-4.1# hdfs dfsadmin -safemode leave
20/01/10 01:57:19 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Safe mode is OFF
bash-4.1#
bash-4.1# hdfs dfs -mkdir /externaltablenew
20/01/10 01:57:27 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
bash-4.1# hdfs dfs -put listings.csv /externaltablenew
20/01/10 01:57:39 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
put: `listings.csv': No such file or directory
bash-4.1# ls
bin  boot  derby.log  dev  etc  home  lib  lib64  media  metastore_db  mnt  opt  proc  resources  root  sbin  selinux  srv  sys  tmp  usr  var
bash-4.1# cd resources/
bash-4.1# ls
listings.csv
bash-4.1# hdfs dfs -put listings.csv /externaltablenew
20/01/10 02:00:09 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
```

```
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 external table dbpdata(
    > id int,
    > name string,
    > host_id string,
    > host_name string,
    > neighbourhood_group string,
    > neighbourhood string,
    > latitude string,
    > longitude string,
    > room_type string,
    > price string,
    > minimum_nights string,
    > number_of_reviews string,
    > last_review string,
    > reviews_per_month string,
    > calculated_host_listings_count string,
    > availability_365 string
    > )row format delimited fields terminated by ',' LOCATION '/externaltablenew/';
OK
Time taken: 0.701 seconds
```

2.1) Analyze the flowing using Hadoop Map Reduce

1. Total number of rentals that are available 365 days a year.

```
bash-4.1# yarn jar booking-0.0.1-SNAPSHOT.jar com.example.map.Available365 listings.csv output/wordCount1n
20/01/12 08:30:10 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
20/01/12 08:30:10 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
20/01/12 08:30:11 INFO input.FileInputFormat: Total input paths to process : 1
20/01/12 08:30:11 INFO mapreduce.JobSubmitter: number of splits:1
20/01/12 08:30:11 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1578830731805_0007
20/01/12 08:30:11 INFO impl.YarnClientImpl: Submitted application application_1578830731805_0007
20/01/12 08:30:11 INFO mapreduce.Job: The url to track the job: http://35b1154b4e48:8088/proxy/application_1578830731805_0007/
20/01/12 08:30:11 INFO mapreduce.Job: Running job: job_1578830731805_0007
20/01/12 08:30:16 INFO mapreduce.Job: Job job_1578830731805_0007 running in uber mode : false
20/01/12 08:30:16 INFO mapreduce.Job:  map 0% reduce 0%
20/01/12 08:30:20 INFO mapreduce.Job:  map 100% reduce 0%
20/01/12 08:30:20 INFO mapreduce.Job: Job job_1578830731805_0007 completed successfully
20/01/12 08:30:21 INFO mapreduce.Job: Counters: 31
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=115435
                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=0
                HDFS: Number of read operations=5
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=1910
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=1910
                Total vcore-seconds taken by all map tasks=1910
                Total megabyte-seconds taken by all map tasks=1955840
        Map-Reduce Framework
                Map input records=7922
                Map output records=0
                Input split bytes=112
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=28
                CPU time spent (ms)=590
                Physical memory (bytes) snapshot=163041280
                Virtual memory (bytes) snapshot=754331648
                Total committed heap usage (bytes)=148897792
        State
                365=843
        File Input Format Counters
                Bytes Read=1164675
        File Output Format Counters
                Bytes Written=0
available for 365 count:        843
bash-4.1#
```

2. Number of rentals per neighbourhood_group

```
bash-4.1# yarn jar booking-0.0.1-SNAPSHOT.jar com.example.map.TotalRentals listings.csv output/wordCount1n
20/01/12 08:14:48 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
20/01/12 08:14:48 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
20/01/12 08:14:49 INFO input.FileInputFormat: Total input paths to process : 1
20/01/12 08:14:49 INFO mapreduce.JobSubmitter: number of splits:1
20/01/12 08:14:49 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1578830731805_0006
20/01/12 08:14:49 INFO impl.YarnClientImpl: Submitted application application_1578830731805_0006
20/01/12 08:14:49 INFO mapreduce.Job: The url to track the job: http://35b1154b4e48:8088/proxy/application_1578830731805_0006/
20/01/12 08:14:49 INFO mapreduce.Job: Running job: job_1578830731805_0006
20/01/12 08:14:53 INFO mapreduce.Job: Job job_1578830731805_0006 running in uber mode : false
20/01/12 08:14:53 INFO mapreduce.Job:  map 0% reduce 0%
20/01/12 08:14:57 INFO mapreduce.Job:  map 100% reduce 0%
20/01/12 08:14:57 INFO mapreduce.Job: Job job_1578830731805_0006 completed successfully
20/01/12 08:14:57 INFO mapreduce.Job: Counters: 35
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=115437
                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=0
                HDFS: Number of read operations=5
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=1833
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=1833
                Total vcore-seconds taken by all map tasks=1833
                Total megabyte-seconds taken by all map tasks=1876992
        Map-Reduce Framework
                Map input records=7922
                Map output records=0
                Input split bytes=112
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=17
                CPU time spent (ms)=650
                Physical memory (bytes) snapshot=162451456
                Virtual memory (bytes) snapshot=753610752
                Total committed heap usage (bytes)=148897792
        Region
                Central Region=5350
                East Region=455
                North Region=168
                North-East Region=295
                West Region=453
        File Input Format Counters
                Bytes Read=1164675
        File Output Format Counters
                Bytes Written=0
```

```
Central Region  5350
East Region     455
North Region    168
North-East Region       295
West Region     453
```

2.2)Analyze the flowing using Hive or Pig

2.2)1. Average price of Private room rental by neighbourhood_group.

```
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>  select neighbourhood_group,avg(cast(price as float)) price_ from dbpdata where room_type='Private room' group by neighbourhood_group;
Query ID = root_20200110021728_c684147c-6581-4983-a113-8c3b4c7157a7
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_1578639354845_0004, Tracking URL = http://42ed2e4b750c:8088/proxy/application_1578639354845_0004/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1578639354845_0004
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-01-10 02:17:35,103 Stage-1 map = 0%,  reduce = 0%
2020-01-10 02:17:39,303 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.25 sec
2020-01-10 02:17:44,424 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.23 sec
MapReduce Total cumulative CPU time: 2 seconds 230 msec
Ended Job = job_1578639354845_0004
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 2.23 sec   HDFS Read: 1175535 HDFS Write: 161 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 230 msec
OK
Central Region  121.04909983633388
East Region     121.28528528528528
North Region    79.7094017094017
North-East Region       79.9531914893617
West Region     125.21913580246914
Time taken: 16.589 seconds, Fetched: 5 row(s)
hive>
```

2.2)2. Top 10 neighbourhood based on Average price of Private room.

```
hive> select neighbourhood ,avg(cast(price as float)) price_ from dbpdata where room_type='Private room' group by neighbourhood order by  price_ desc limit 10;
Query ID = root_20200110021946_232437ff-7484-4b18-a47d-397460cb734b
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_1578639354845_0005, Tracking URL = http://42ed2e4b750c:8088/proxy/application_1578639354845_0005/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1578639354845_0005
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-01-10 02:19:52,223 Stage-1 map = 0%,  reduce = 0%
2020-01-10 02:19:56,342 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.22 sec
2020-01-10 02:20:01,457 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.1 sec
MapReduce Total cumulative CPU time: 2 seconds 100 msec
Ended Job = job_1578639354845_0005
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_1578639354845_0006, Tracking URL = http://42ed2e4b750c:8088/proxy/application_1578639354845_0006/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1578639354845_0006
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2020-01-10 02:20:09,943 Stage-2 map = 0%,  reduce = 0%
2020-01-10 02:20:14,102 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 0.55 sec
2020-01-10 02:20:18,247 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 1.43 sec
MapReduce Total cumulative CPU time: 1 seconds 430 msec
Ended Job = job_1578639354845_0006
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 2.1 sec   HDFS Read: 1175106 HDFS Write: 1566 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 1.43 sec   HDFS Read: 6316 HDFS Write: 283 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 530 msec
OK
Southern Islands        649.6666666666666
Bukit Panjang   422.67857142857144
Marina South    419.0
Jurong East     192.65
Downtown Core   169.79775280898878
Toa Payoh       153.38636363636363
Singapore River 151.92307692307693
Orchard 149.17948717948718
Bishan  146.42857142857142
Kallang 146.13813813813815
Time taken: 32.405 seconds, Fetched: 10 row(s)
```

2.2)3. The 5 lowest price properties per each room_type.

i. Private room Type

```
hive> select host_id,host_name , cast(price as float) price_ from dbpdata where room_type='Private room' order by  price_ desc limit 5;
Query ID = root_20200110022613_5ef3f8a4-ef4d-46e7-bad6-df839eb88ba6
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_1578639354845_0007, Tracking URL = http://42ed2e4b750c:8088/proxy/application_1578639354845_0007/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1578639354845_0007
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-01-10 02:26:17,194 Stage-1 map = 0%,  reduce = 0%
2020-01-10 02:26:21,337 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.39 sec
2020-01-10 02:26:26,521 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.45 sec
MapReduce Total cumulative CPU time: 2 seconds 450 msec
Ended Job = job_1578639354845_0007
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 2.45 sec   HDFS Read: 1173931 HDFS Write: 111 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 450 msec
OK
68223771        Yolivia 10000.0
20307016        David   10000.0
60916030        Yin     7000.0
139204582       X-Roy   7000.0
188629774       Jo      6000.0
Time taken: 13.978 seconds, Fetched: 5 row(s)
```

ii. Shared Room Type

```
hive> select host_id,host_name , cast(price as float) price_ from dbpdata where room_type='Shared room' order by  price_ desc limit 5;
Query ID = root_20200110022808_2317dd42-3f6a-41c7-a0ed-8797a3432dad
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_1578639354845_0008, Tracking URL = http://42ed2e4b750c:8088/proxy/application_1578639354845_0008/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1578639354845_0008
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-01-10 02:28:12,854 Stage-1 map = 0%,  reduce = 0%
2020-01-10 02:28:16,959 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.17 sec
2020-01-10 02:28:21,042 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.09 sec
MapReduce Total cumulative CPU time: 2 seconds 90 msec
Ended Job = job_1578639354845_0008
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 2.09 sec   HDFS Read: 1173929 HDFS Write: 111 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 90 msec
OK
41059478        Andrew  2500.0
45863208        Sean    1100.0
28837767        Jia Yi  1000.0
180974037       Vikar   600.0
91255816        Xueqin  500.0
Time taken: 13.717 seconds, Fetched: 5 row(s)
```

iii. Entire home/apt

```
hive> select host_id,host_name , cast(price as float) price_ from dbpdata where room_type='Entire home/apt' order by  price_ desc limit 5;
Query ID = root_20200110022942_cf702368-8a49-435f-a0fd-b73586b97e91
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_1578639354845_0009, Tracking URL = http://42ed2e4b750c:8088/proxy/application_1578639354845_0009/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1578639354845_0009
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-01-10 02:29:46,042 Stage-1 map = 0%,  reduce = 0%
2020-01-10 02:29:51,220 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.53 sec
2020-01-10 02:29:55,338 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.64 sec
MapReduce Total cumulative CPU time: 2 seconds 640 msec
Ended Job = job_1578639354845_0009
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 2.64 sec   HDFS Read: 1173937 HDFS Write: 113 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 640 msec
OK
122991242       Jj      10000.0
34411185        Darren  8900.0
34411185        Darren  8900.0
106682987       Lily    6944.0
5654051 Jonathan        5000.0
Time taken: 13.863 seconds, Fetched: 5 row(s)
```

2.3)

Analyze the flowing using Spark

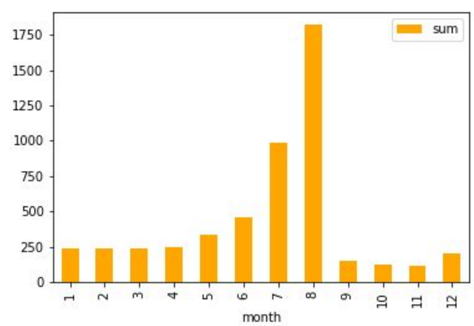
2.3)1. Percentage of owners who rent more than one property.

```
%pyspark

#2.3 .
#1. Percentage of owners who rent more than one property.

from pyspark.sql import *

#DF as a SQL tem view
df.createOrReplaceTempView("listing_temp_tbl")
#df.show()
#Fetch the count of tot owners
owners = spark.sql("select host_id, host_name, COUNT(*) as Properties from listing_temp_tbl group by host_id, host_name Order by Properties DESC")
OwnersCount=owners.count()

# Getting the count of tot no of multiple property owners
ownersgtone = spark.sql("select host_id, host_name, COUNT(*) as Properties from listing_temp_tbl group by host_id, host_name HAVING Properties>1 Order by Properties DESC")
ownersgtone_1=ownersgtone.count()
percentageOfMultipleOwners = (float(ownersgtone_1)/OwnersCount)*100.0
print("***Percentage of owners who rent more than one property***")
print(percentageOfMultipleOwners)

***Percentage of owners who rent more than one property***
27.3163528977
```

2.3)2. Histogram of number of rentals reviewed over time (based on last_review) in mouth Granularity.

```
%pyspark
#2.3
#2. Histogram of number of rentals reviewed over time (based on last_review) in mouth granularity.
import pandas as pd
from pyspark.sql.functions import *
histogramdf = spark.sql("Select month(last_review)month  ,count(*)as sum from listing_temp_tbl where month(last_review) is not null group by month order by month")
print(histogramdf.show())

histogramdfpandas = histogramdf.toPandas()
histPlot = histogramdfpandas.plot.bar(x='month', y='sum', color='orange')
```

```
|month| sum|
+-----+----+
|    1| 241|
|    2| 236|
|    3| 235|
|    4| 249|
|    5| 334|
|    6| 455|
|    7| 983|
|    8|1822|
|    9| 149|
|   10| 125|
|   11| 111|
|   12| 201|
+-----+----+

None
<matplotlib.figure.Figure at 0x7fcd0c6dc8d0>
```



Took 2 sec. Last updated by anonymous at January 10 2020, 4:47:36 PM.

2.3)3. 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.

SPARK JOBS FINISHED

```
%pyspark

#2.3
#3. 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.
top5Arentals = spark.sql("Select * from listing_temp_tbl  where  cast(availability_365 as string)like '%365%'")
top5Arentals.createOrReplaceTempView("listingtemptable_365")
df1=spark.sql(" select * from listingtemptable_365 ")
#df1.show()
top5Arentals1 = spark.sql("Select neighbourhood_group, neighbourhood, count(*) as count from listingtemptable_365 where neighbourhood in (Select neighbourhood  From listingtemptable_365 Group By neighbourhood_group, neighbourhood  order By avg(price) Limit 5)  group by neighbourhood_group,
    neighbourhood")

top5Arentals1.show()
```

```
+-------------------+--------------------+-----+
|neighbourhood_group|       neighbourhood|count|
+-------------------+--------------------+-----+
|        West Region|Western Water Cat...|    1|
|        West Region|       Bukit Panjang|    1|
|       North Region|           Woodlands|   12|
|  North-East Region|           Serangoon|    8|
|       North Region|        Lim Chu Kang|    1|
+-------------------+--------------------+-----+
```

3. Performing Machine Learning on the Singapore Airbnb Data

Import required libraries

```
%pyspark

# 3.ML on the Singapore Airbnb Data

from pyspark.sql import SparkSession, Row, functions, types
from pyspark.sql.functions import udf
import numpy as np
import pandas as pd
from pandas import DataFrame
from pyspark.ml.feature import HashingTF, IDF, Tokenizer, VectorAssembler, StringIndexer
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.mllib.regression import LabeledPoint
from pyspark.ml.classification import NaiveBayes
from pyspark.mllib.evaluation import MultilabelMetrics
from pyspark.mllib.linalg import Vectors
from pyspark.ml import Pipeline
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
from pyspark.mllib.evaluation import MulticlassMetrics, BinaryClassificationMetrics
from pyspark.rdd import RDD
from pyspark.sql.functions import col
import pandas as pd
import numpy as np
from pyspark.sql import functions as F
from pyspark.sql.functions import when
from pyspark.sql import SparkSession
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.classification import LogisticRegression
from pyspark.mllib.evaluation import MulticlassMetrics
from pyspark.sql import functions as F
from pyspark.sql.functions import lit
from datetime import date
from pyspark.sql.types import StructField
from pyspark.sql.types import StructType
from pyspark.sql.types import *
from pyspark.mllib.evaluation import MulticlassMetrics
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.mllib.evaluation import MultilabelMetrics
from pyspark.mllib.evaluation import MulticlassMetrics, BinaryClassificationMetrics
from pyspark.rdd import RDD
```

Next, we have to do the preprocessing and make our training and testing data sets.

```
%pyspark
#3.Data Preprocessing
#COlumns need tor the model tarining and make a new dataframe
features=['id','latitude','longitude','neighbourhood_group']
df_training = df.select(features)
#df_training.show()
df_training = df_training.na.drop()
df_training = df_training.withColumn("longitude", df_training["longitude"].cast(DoubleType()))
df_training.printSchema()

df_training.createOrReplaceTempView("listings_view")

df1_training = spark.sql("select  neighbourhood_group, count(id) id_count from listings_view group by neighbourhood_group order by id_count DESC")
#df1_training.show()
#Ignore the neighbourhood group in which they have count less than 10

def valueToInt(value):
    if value=='Central Region': return 1
    elif value=='West Region': return  2
    elif value=='East Region': return 3
    elif value=='North-East Region': return 4
    elif value=='North Region': return 5
    else: return 6

udfValueToInt = udf(valueToInt, IntegerType())
df1_training = df_training.withColumn("label_column", udfValueToInt("neighbourhood_group"))
df2_training = df1_training.filter(df1_training['label_column'] < 6)
df2_training.show()
```

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

+------+--------+---------+-------------------+------------+
|    id|latitude|longitude|neighbourhood_group|label_column|
+------+--------+---------+-------------------+------------+
| 49091| 1.44255| 103.7958|       North Region|           5|
| 50646| 1.33235|103.78521|     Central Region|           1|
| 56334| 1.44246|103.79667|       North Region|           5|
| 71609| 1.34541|103.95712|        East Region|           3|
| 71896| 1.34567|103.95963|        East Region|           3|
| 71903| 1.34702|103.96103|        East Region|           3|
| 71907| 1.34348|103.96337|        East Region|           3|
|241503| 1.32304|103.91363|        East Region|           3|
```

Created the feature vector from dataset and dividing them into training and testing data(70% for training and 30% for testing) after scaling data.

```
%pyspark
def vectAssembler(df,impFeatures):
    print("Vector assembler ")
    assembler = VectorAssembler(inputCols=impFeatures,outputCol="features")
    df_new=assembler.transform(df)
    print("vector assembling is done.")
    return df_new
impFeatures=['latitude','longitude']
impdf = df2_training.select(impFeatures).show()
# Create a feature vector column using latitude and longitude
dtf=vectAssembler(df2_training,impFeatures)

finalized_data = dtf.select('label_column', 'features')
from pyspark.ml.feature import StandardScaler
scaler = StandardScaler(inputCol='features', outputCol='scaledFeatures', withStd=True, withMean=True)
scalerModel = scaler.fit(finalized_data)
classiFinalData = scalerModel.transform(finalized_data)

# Splitting data into training and test sets (30% for testing)
(trainingData, testData) = classiFinalData.randomSplit([0.7, 0.3])

trainingData.show()
```

```
+--------+---------+
only showing top 20 rows


Vector assembler
vector assembling is done.
+------------+------------------+--------------------+
|label_column|          features|      scaledFeatures|
+------------+------------------+--------------------+
|           1|[1.24387,103.84246]|[-2.3040593559511...|
|           1|[1.24526,103.83999]|[-2.2585108654236...|
|           1|[1.24847,103.82389]|[-2.1533233441333...|
|           1|[1.24853,103.82502]|[-2.1513572222400...|
|           1|[1.24881,103.82364]|[-2.1421819867381...|
|           1|[1.24918,103.82509]|[-2.1300575683962...|
|           1|[1.24992,103.82441]|[-2.1058087317125...|
|           1| [1.2504,103.82539]|[-2.0900797565663...|
|           1|[1.25046,103.82529]|[-2.0881136346730...|
```

Took 1 sec. Last updated by anonymous at January 10 2020, 2:49:07 PM. (outdated)

After train and testing, get the predictions.

```
%pyspark
def LRTraining(trainDF):
    print("Model fitting started")
    lr = LogisticRegression(labelCol="label_column", featuresCol="scaledFeatures",elasticNetParam=0.8, family="multinomial",maxIter=120)
    model=lr.fit(trainDF)
    print("Model fitting completed")
    return model

model = LRTraining(trainingData)

# Get predictions.
predictions = model.transform(testData)

# Predicted data
predictions.select("prediction", "label_column", "features").show(5)

Model fitting started
Model fitting completed
+----------+------------+------------------+
|prediction|label_column|          features|
+----------+------------+------------------+
|       1.0|           1|[1.24391,103.83915]|
|       1.0|           1|[1.25284,103.82225]|
|       1.0|           1|[1.25639,103.82302]|
|       1.0|           1|[1.26478,103.81762]|
|       1.0|           1| [1.2667,103.81127]|
+----------+------------+------------------+
only showing top 5 rows
```

Model Evaluation

```
%pyspark
from pyspark.mllib.evaluation import MulticlassMetrics
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.mllib.evaluation import MultilabelMetrics
from pyspark.mllib.evaluation import MulticlassMetrics, BinaryClassificationMetrics
from pyspark.rdd import RDD

evaluatorRecall = MulticlassClassificationEvaluator(labelCol="label_column", predictionCol="prediction", metricName="weightedRecall")
evaluatorPrecision = MulticlassClassificationEvaluator(labelCol="label_column", predictionCol="prediction", metricName="weightedPrecision")
recall = evaluatorRecall.evaluate(predictions)
precision = evaluatorPrecision.evaluate(predictions)
print("Recall %s" % recall)
print("Precision %s" % precision)

Recall 0.981865284974
Precision 0.981982630764
```

4). Presentation of the analysis for Singapore Airbnb Data

4)1. Total number of rentals that are available 365 days a year, and the total number of rentals.(As numbers)

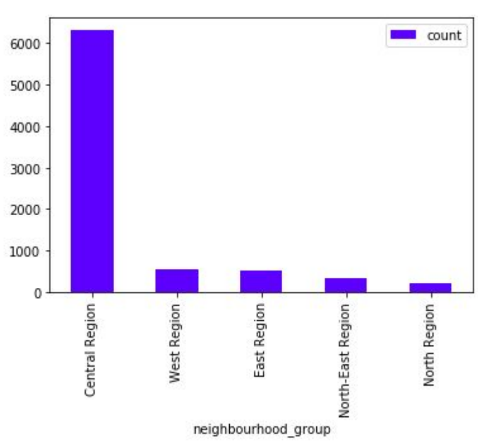
```
%pyspark

#4. Presentation of the analysis for Singapore Airbnb Data
#1.  Total number of rentals that are available 365 days a year, and the total number of rentals.(As numbers)
df_tot_rental=spark.sql("select count(*)totalrentals from  listing_temp_tbl ")
df_tot_rental.show()
df_df_tot_rental_av_365=spark.sql("select count(*)totalrentals_ava_365 from  listing_temp_tbl where availability_365=365 ")
df_df_tot_rental_av_365.show()
```

```
+------------+
|totalrentals|
+------------+
|        7921|
+------------+

+--------------------+
|totalrentals_ava_365|
+--------------------+
|                 843|
+--------------------+
```

4)2. Number of rentals per neighbourhood_group (As a bar chart)

```
%pyspark
#2.  Number of rentals per neighbourhood_group (As a bar chart)

df_rentalsperneighbourhd=spark.sql("select neighbourhood_group,count(*)count  from  listing_temp_tbl group by neighbourhood_group  order by count desc limit 5")
df_rentalsperneighbourhd.show()

histogramdfpandas = df_rentalsperneighbourhd.toPandas()
histPlot = histogramdfpandas.plot.bar(x='neighbourhood_group', y='count', color='b')
```

```
+-------------------+-----+
|neighbourhood_group|count|
+-------------------+-----+
|     Central Region| 6301|
|        West Region|  539|
|        East Region|  508|
|  North-East Region|  344|
|       North Region|  203|
+-------------------+-----+

<matplotlib.figure.Figure at 0x7fcd0e1eec90>
```

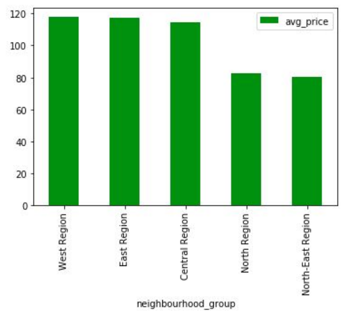


4)3. Average price of Private room rental by neighbourhood_group (As a bar chart).

```
%pyspark
#3 Average price of private room by neighbourhood group.
df_avgpricebyneibourgoodgroup=spark.sql("select neighbourhood_group,avg(cast(price as float))avg_price  from  listing_temp_tbl where room_type = 'Private room' group by neighbourhood_group  order by avg_price desc limit 5")
df_avgpricebyneibourgoodgroup.show()
histogramdfpandas = df_avgpricebyneibourgoodgroup.toPandas()
histPlot = histogramdfpandas.plot.bar(x='neighbourhood_group', y='avg_price', color='g')
```

neighbourhood_group	avg_price
West Region	117.82539682539682
East Region	117.23497267759562
Central Region	114.47408742676882
North Region	82.35460992907801
North-East Region	80.06296296296296

```
<matplotlib.figure.Figure at 0x7fcd0e05e9d0>
```



Took 2 sec. Last updated by anonymous at January 10 2020, 3:52:05 PM.

4)4. Top 10 neighbourhood based on Average price of Private room (As a table).

```
%pyspark

#4 Top 10 neighbourhood based on Average price of Private room
df_avgpricebyneibourhood=spark.sql("select neighbourhood_group, neighbourhood,avg(cast(price as float))avg_price  from  listing_temp_tbl where room_type = 'Private room' group by neighbourhood_group,neighbourhood  order by avg_price desc limit 10")
df_avgpricebyneibourhood.show()
```

neighbourhood_group	neighbourhood	avg_price
Central Region	Southern Islands	649.6666666666666
Central Region	Marina South	419.0
West Region	Bukit Panjang	409.44827586206895
West Region	Jurong East	182.25757575757575
Central Region	Downtown Core	163.5047619047619
Central Region	Singapore River	150.66666666666666
Central Region	Orchard	146.89795918367346
Central Region	Toa Payoh	142.78
Central Region	Bishan	138.92105263157896
Central Region	Outram	135.26639344262296

4)5. The 5 lowest price rentals per each room_type (As a table).

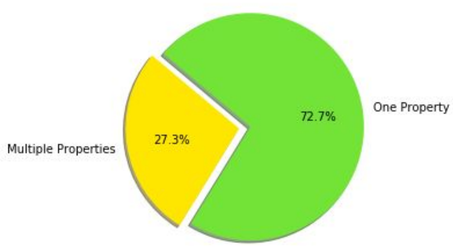
```
%pyspark
# 5 lowest price per each room type
df_avgpricebyneibourhood=spark.sql("select neighbourhood_group, neighbourhood,price  from  listing_temp_tbl where room_type = 'Private room' order by price asc limit 5")
df_avgpricebyneibourhood.show()
df_avgpricebyneibourhood=spark.sql("select neighbourhood_group, neighbourhood,price  from  listing_temp_tbl where room_type = 'Shared room' order by price asc limit 5")
df_avgpricebyneibourhood.show()
df_avgpricebyneibourhood=spark.sql("select neighbourhood_group, neighbourhood,price  from  listing_temp_tbl where room_type = 'Entire home/apt' order by price asc limit 5")
df_avgpricebyneibourhood.show()
```

```
+-------------------+-------------+-----+
|neighbourhood_group|neighbourhood|price|
+-------------------+-------------+-----+
|     Central Region|Marine Parade|   14|
|     Central Region|      Geylang|   15|
|     Central Region|       Outram|   15|
|        East Region|     Tampines|   15|
|        West Region|  Jurong West|   15|
+-------------------+-------------+-----+

+-------------------+-------------+-----+
|neighbourhood_group|neighbourhood|price|
+-------------------+-------------+-----+
|        East Region|        Bedok|   14|
|        West Region|  Jurong West|   15|
|     Central Region|      Kallang|   18|
|     Central Region|       Rochor|   18|
|     Central Region|      Kallang|   19|
+-------------------+-------------+-----+

+-------------------+-------------+-----+
|neighbourhood_group|neighbourhood|price|
+-------------------+-------------+-----+
|     Central Region|       Rochor|    0|
|     Central Region|      Geylang|   14|
|        West Region|Bukit Panjang|   14|
|     Central Region|  Bukit Timah|   31|
|       North Region|       Yishun|   39|
+-------------------+-------------+-----+
```

4)6. Percentage of owners who rent more than one property (As a pie chart).

```
%pyspark
import matplotlib.pyplot as plt
#Percentage of owners who rent more than one property

# Getting the count of total number of owners
owners = spark.sql("select host_id, host_name, COUNT(*) as Properties from listing_temp_tbl group by host_id, host_name Order by Properties DESC")
#print(owners.show())
##Get the owners count
OwnersCount=owners.count()


# Getting the count of total number of multiple property owners
ownersgtone = spark.sql("select host_id, host_name, COUNT(*) as Properties from listing_temp_tbl group by host_id, host_name HAVING Properties>1 Order by Properties DESC")
ownersgtone_1=ownersgtone.count()
percentageOfMultipleOwners = (float(ownersgtone_1)/OwnersCount)*100.0
print("##########Percentage of owners who rent more than one property#################")
print(percentageOfMultipleOwners)


# Data to plot
labels = 'Multiple Properties', 'One Property'
sizes = [percentageOfMultipleOwners, 100-percentageOfMultipleOwners, ]
colors = ['gold', 'yellowgreen']
explode = (0.1, 0)  # explode 1st slice

# Plot
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=140)

plt.axis('equal')
plt.show()
```

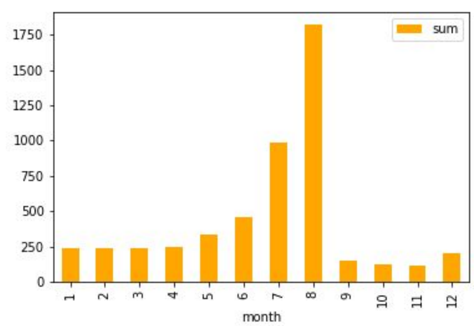
```
##########Percentage of owners who rent more than one property#################
27.3163528977
<matplotlib.figure.Figure at 0x7fcd0c62a4d0>
```



4)7. Histogram of number of rentals reviewed over time (based on last_review) in mouth granularity (As a bar chart).

```
%pyspark
import pandas as pd
from pyspark.sql.functions import *
histogramdf = spark.sql("Select month(last_review)month  ,count(*)as sum from listing_temp_tbl where month(last_review) is not null group by month order by month")
print(histogramdf.show())

histogramdfpandas = histogramdf.toPandas()
histPlot = histogramdfpandas.plot.bar(x='month', y='sum', color='orange')
```

```
+-----+----+
|month| sum|
+-----+----+
|    1| 241|
|    2| 236|
|    3| 235|
|    4| 249|
|    5| 334|
|    6| 455|
|    7| 983|
|    8|1822|
|    9| 149|
|   10| 125|
|   11| 111|
|   12| 201|
+-----+----+

None
```



Took 2 sec. Last updated by anonymous at January 10 2020, 4:49:02 PM. (outdated)

4)8. 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 (As a table).

SPARK JOBS FINISHED

```
%pyspark
top5Arentals = spark.sql("Select * from listing_temp_tbl  where  cast(availability_365 as string)like '%365%'")
top5Arentals.createOrReplaceTempView("listingtemptable_365")
df1=spark.sql(" select * from listingtemptable_365 ")
#df1.show()
top5Arentals1 = spark.sql("Select neighbourhood_group, neighbourhood, count(*) as count from listingtemptable_365 where neighbourhood in (Select neighbourhood  From listingtemptable_365 Group By neighbourhood_group, neighbourhood  order By avg(price) Limit 5)  group by neighbourhood_group,
    neighbourhood")

top5Arentals1.show()
```

```
+-----------------+--------------------+-----+
|neighbourhood_group|       neighbourhood|count|
+-----------------+--------------------+-----+
|      West Region|Western Water Cat...|    1|
|      West Region|       Bukit Panjang|    1|
|     North Region|           Woodlands|   12|
| North-East Region|           Serangoon|    8|
|     North Region|        Lim Chu Kang|    1|
+-----------------+--------------------+-----+
```

Following is the web dashboard created to represent the data.

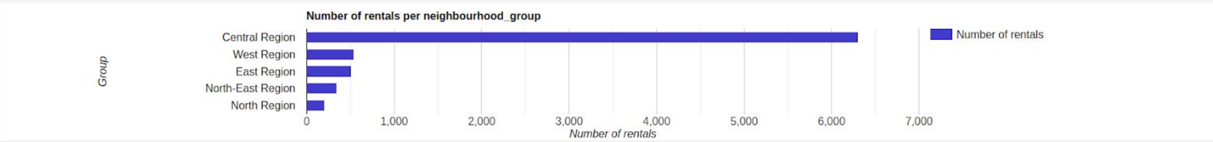
Singapore Airbnb Data

4)1. Total number of rentals that are available 365 days a year, and the total number of rentals.(As numbers)

Rentals that are available 365 days a year = 843

Total number of Rentals = 7921

4)2. Number of rentals per neighbourhood_group (As a bar chart)



4)3. Average price of Private room rental by neighbourhood_group (As a bar chart)

4)4. Top 10 neighbourhood based on Average price of Private room (As a table)

No.	Neighbourhood group	Neighbourhood	Average price
1	Central Region	Southern Islands	649.67
2	Central Region	Marina South	419.0
3	West Region	Bukit Panjang	409.45
4	West Region	Jurong East	182.26
5	Central Region	Downtown Core	163.50
6	Central Region	Singapore River	150.67
7	Central Region	Orchard	146.90
8	Central Region	Toa Payoh	142.78
9	Central Region	Bishan	138.92
10	Central Region	Outram	135.27

4)5. The 5 lowest price rentals per each room_type (As a table)

Private Room

No.	id	host id	host name	Neighbourhood group	Neighbourhood	Price
1	18679631	108408404	Sutthida	Central Region	Marine Parade	14
2	21926382	45343820	Deqing	East Region	Tampines	15
3	33324090	13503463	Angelina	West Region	Jurong West	15
4	10318835	13460992	Ming	Central Region	Geylang	15
5	24883645	14021375	Marc	Central Region	Outram	15

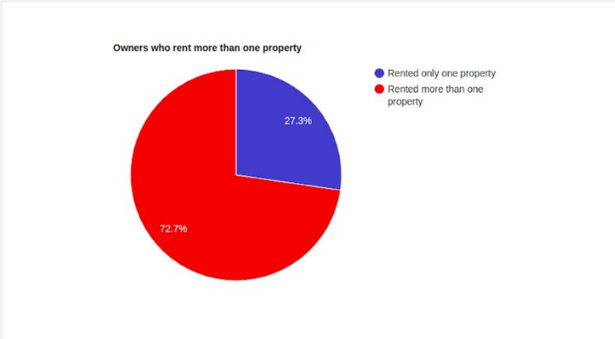
Entire home/apt

No.	id	host id	host name	Neighbourhood group	Neighbourhood	Price
1	21408571	114674497	Mitul	Central Region	Rochor	0
2	37506711	29799617	John	Central Region	Geylang	14
3	35947264	75175440	Rain	West Region	Bukit Panjang	14
4	16381367	26246420	Jordan	Central Region	Bukit Timah	31
5	36761101	73254645	Jennifer	North Region	Yishun	39

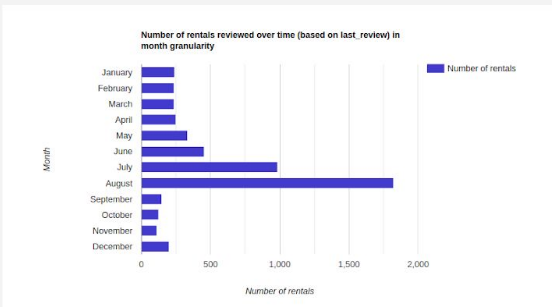
Shared room

No.	id	host id	host name	Neighbourhood group	Neighbourhood	Price
1	18656726	21900076	Mary	East Region	Bedok	14
2	22034488	160839396	Bi	West Region	Jurong West	15
3	26170424	196709892	Anna	Central Region	Rochor	18
4	10040828	46545593	Meadows	Central Region	Kallang	18
5	20839977	63448912	River City Inn	Central Region	Singapore River	19

4)6. Percentage of owners who rent more than one property (As a pie chart)



4)7. Histogram of number of rentals reviewed over time (based on last_review) in mouth granularity (As a bar chart)



4)8. 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 (As a table)

No.	Neighbourhood group	Neighbourhood	Average price
1	West Region	Western Water Catchment	1
2	West Region	Bukit Panjang	1
3	North Region	Woodlands	12
4	North-East Region	Serangoon	8
5	North Region	Lim Chu Kang	1

[REFERENCES]

[1]https://www.xenonstack.com/blog/big-data-ingestion/