Cloud Computing Capstone Task 1 Report- Sharad Narang

- Give a brief overview of how you extracted and cleaned the data.

Data Extraction & Cleansing of the transportation was done with following steps

1. Moving of EBS snapshot to local region Volume creation from Snapshot
    a. EBS Snapshot for the transportation dataset was provided as **snap-e1608d88** for Linux/Unix, it was copied from us-east-1 (N. Virginia) region to **US West (Oregon) region where my Hadoop and Spark instances were running.**
    b. Restored an Amazon EBS Volume from the Snapshot – Referred following link: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-restoring-volume.html
2. Attached the concerned Amazon EBS Volume to Hadoop Master Node Instance - Snapshot – Referred following link : http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-attaching-volume.html
3. Made the Amazon EBS Volume Available for Use - Referred following link: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-using-volumes.html
    i. sudo file -s /dev/xvdf
    ii. sudo mount /dev/xvdi /final_data
    iii. Made entry into /etc/fstab to mount this EBS volume on every system reboot
4. Analyzed the data and the given data requirements in the questions. Figured out the aviation data and specifically airlie_ontime data will be needed for the project.
5. Preparing the data – unzipped all the airline_ontime data recursively and put in local directory – Following is the code extract

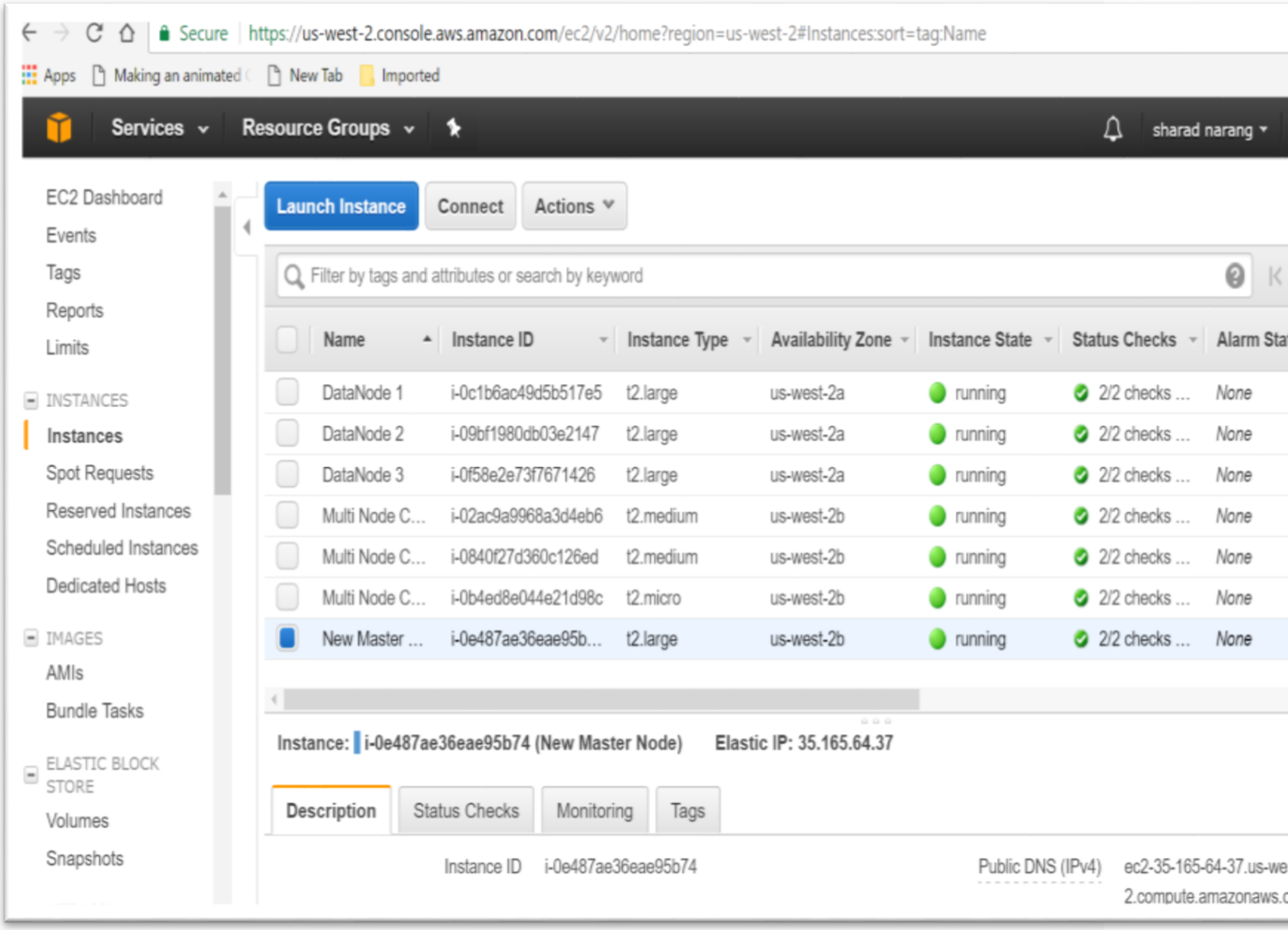
```
rootdir = '/final_data/aviation/airline_ontime/'

tgt_dir_name = '/home/ubuntu/capstone_data'

extension = ".zip"

for subdir, dirs, files in os.walk(rootdir):

    for file in files:

        file_name = os.path.join(subdir, file)

        zip_ref = zipfile.ZipFile(file_name) # create zipfile object

        zip_ref.extractall(tgt_dir_name) # extract file to dir

        zip_ref.close() # close file
```

6. Loading into HDFS – Made the directory and copied all the csv files into HDFS
    a. Hdfs dfs -mkdir /user
    b. hdfs dfs -copyFromLocal capstone_data/*csv /user

7. Cleaning & Extracting Through Spark – Analyzed all the data requirements for the given questions and came up with the list of selected fields from airline on time data, used spark program to read the selective attributes and metrics from the data set and stored into more optimal parquet format from storage and processing perspective
    a. df = sqlContext.read.load('hdfs:///user/*csv', format='com.databricks.spark.csv', header='true', inferSchema='true',treatEmptyValuesAsNulls= 'true',nullValue="")
    b. df.registerTempTable("ontime")
    c. result = sqlContext.sql("SELECT Year,Quarter,Month,DayofMonth,DayOfWeek,FlightDate,FlightNum,CRSDepTime,DepTime,UniqueCarrier,AirlineID,Carrier,Origin,OriginCityName,OriginState,Dest,DestCityName,DestState,DestStateName,DepDelayMinutes,DepDelay,ArrDelay,ArrDelayMinutes,Cancelled FROM ontime")
    d. result.write.parquet("/user/final_ontime.parquet")
8. Clean Up of HDFS – Besides the generated parquet file removed all the other data files from HDFS
    a. hdfs dfs -rmr/user/*csv

- Give a brief overview of how you integrated each system.

    - Spin up AWS EC2 Instances– Created the 4 EC2 large Ubuntu machines, treating one as the NameNode(master) and the remaining three as DataNodes. Configured the security group for the exercise. Referred following link: https://blog.insightdatascience.com/spinning-up-a-free-hadoop-cluster-step-by-step-c406d56bae42
    - Also added a 3 node multi node machines for setting up Cassandra Cluster



The NameNode in the Hadoop cluster needs to be able to communicate with the other DataNodes in the cluster. This is done by configuring passwordless SSH between the NameNode and the DataNodes.

```
$ cat ~/.ssh/id_rsa.pub | ssh datanode1 'cat >> ~/.ssh/authorized_keys'
$ cat ~/.ssh/id_rsa.pub | ssh datanode2 'cat >> ~/.ssh/authorized_keys'
$ cat ~/.ssh/id_rsa.pub | ssh datanode3 'cat >> ~/.ssh/authorized_keys'
```

- HDFS Set Up & Configuration - . Referred following link: https://blog.insightdatascience.com/spinning-up-a-free-hadoop-cluster-step-by-step-c406d56bae42
    - Install Java sudo apt-get update & sudo apt-get install openjdk-7-jdk
    - Install Hadoop
        - wget http://apache.mirrors.tds.net/hadoop/common/hadoop-2.7.1/hadoop-2.7.1.tar.gz -P ~/Downloads
        - sudo tar zxvf ~/Downloads/hadoop-* -C /usr/local
        - sudo mv /usr/local/hadoop-* /usr/local/Hadoop
    - Setting environment Variables

        - export JAVA_HOME=/usr
        - export PATH=$PATH:$JAVA_HOME/bin
        - export HADOOP_HOME=/usr/local/hadoop
        - export PATH=$PATH:$HADOOP_HOME/bin
        - export HADOOP_CONF_DIR=/usr/local/hadoop/etc/Hadoop
    - Hadoop Configurations – Made the changes on Public DNS for name node in following files
        - $HADOOP_CONF_DIR/hadoop-env.sh
        - $HADOOP_CONF_DIR/core-site.xml
        - $HADOOP_CONF_DIR/yarn-site.xml
        - $HADOOP_CONF_DIR/mapred-site.xml
    - NameNode Specific Configurations
        - adding hosts to /etc/hosts
        - modifying the configurations in $HADOOP_CONF_DIR/hdfs-site.xml
        - defining the Hadoop master in $HADOOP_CONF_DIR/masters –
            - ubuntu@ip-172-31-20-213:~/capstone_code$ cat $HADOOP_CONF_DIR/mastersec2-35-165-64-37.us-west-2.compute.amazonaws.com
        - defining the Hadoop slaves in $HADOOP_CONF_DIR/slaves - ubuntu@ip-172-31-20-213:~/capstone_code$ cat $HADOOP_CONF_DIR/slaves
            - ec2-52-24-126-67.us-west-2.compute.amazonaws.com
            - ec2-34-210-50-210.us-west-2.compute.amazonaws.com
            - ec2-34-210-194-87.us-west-2.compute.amazonaws.com
    - DataNode Specific Configurations - $HADOOP_CONF_DIR/hdfs-site.xml data directory configuration
    - Run the services - $HADOOP_HOME/sbin/start-dfs.sh & $HADOOP_HOME/sbin/start-yarn.sh
- Spark Set Up & configuration Referred following link: http://blog.insightdatalabs.com/spark-cluster-step-by-step/
    - Install Spark –
        - sudo apt-get install scala
        - wget http://apache.mirrors.tds.net/spark/spark-2.0.1/spark-2.0.1-bin-hadoop2.7.tgz -P ~/Downloads
        - sudo tar zxvf ~/Downloads/spark-* -C /usr/local
    - Environment Variables –
        - export SPARK_HOME=/usr/local/spark
        - export PATH=$PATH:$SPARK_HOME/bin
    - Common Spark Configurations on all Nodes - $SPARK_HOME/conf/spark-env.sh
        - ubuntu@ip-172-31-20-213:~/capstone_code$ head -5 $SPARK_HOME/conf/spark-env.sh
        - #!/usr/bin/env bash
        - export JAVA_HOME=/usr
        - export SPARK_PUBLIC_DNS="ec2-35-165-64-37.us-west-2.compute.amazonaws.com"
        - export SPARK_WORKER_CORES=120

        - Spark Master Specific Configurations : ubuntu@ip-172-31-20-213:~/capstone_code$ cat $SPARK_HOME/conf/slaves
            - ec2-52-24-126-67.us-west-2.compute.amazonaws.com
            - ec2-34-210-50-210.us-west-2.compute.amazonaws.com
            - ec2-34-210-194-87.us-west-2.compute.amazonaws.com
        - Start Spark Cluster $SPARK_HOME/sbin/start-all.sh
- Cassandra Set Up & configuration Referred Link : http://datascale.io/how-to-create-a-cassandra-cluster-in-aws-part-2/
    - Java Version Update
        - $ sudo add-apt-repository ppa:webupd8team/java
        - $ sudo apt-get update
        - $ sudo apt-get install oracle-java8-installer
    - Install Cassandra
        - Add the repository to the `/etc/apt/sources.list.d/cassandra.sources.list` -echo "deb http://debian.datastax.com/community stable main" | sudo tee -a /etc/apt/sources.list.d/cassandra.sources.list
        - Add the DataStax repository key to your aptitude trusted keys. vim /etc/apt/sources.list adding - deb http://some.debian.mirror/debian/ $distro main contrib non-free
        - curl -L https://debian.datastax.com/debian/repo_key | sudo apt-key add -
        - Install the latest package:
            - $ sudo apt-get update
            - $ sudo apt-get install dsc30
            - $ sudo apt-get install cassandra-tools
    - Configure Cassandra
        - Stop Cassandra: sudo service cassandra stop & Clean the data sudo rm -rf /var/lib/cassandra/data/system/*
        - Set the properties in the cassandra.yaml file for each node
            - cluster_name: Demo Cluster
            - num_tokens: recommended value: 256
            - -seeds: internal public IP address of each seed node : "54.190.206.69"
            - listen_address:Private IP 172.31.24.183
            - rpc_address:listen address for client connections
            - endpoint_snitch: SimpleSnitch
        - Bring up the cluster
            - $ sudo service cassandra stop
            - $ sudo rm -rf /var/lib/cassandra/data/system/*
            - $ sudo service cassandra start
- Integration of Spark & Hadoop - HADOOP_CONF_DIR or YARN_CONF_DIR points to the directory which contains the (client side) configuration files for the Hadoop cluster. These configs are used to write to HDFS and connect to the YARN ResourceManager.
  Set HADOOP_CONF_DIR in $SPARK_HOME/spark-env.sh to a location containing the configuration files
- Integration of Cassandra & Spark – Used the following Datastax - Spark & Cassandra connector Refer Link : https://github.com/datastax/spark-cassandra-connector

- Based on Version Compatibility – pyspark/Spark-submit --packages datastax:spark-cassandra-connector:2.0.1-s_2.11 --conf spark.cassandra.connection.host=54.218.50.19

- What approaches and algorithms did you use to answer each question?

  - Used SPARKSQL Library for getting the data for each of the question, Read the consolidated parquet file generated by merging all csv files for airline ontime data ranging from year 1988 to 2008 and with selected attributes and metrics .Registered it as a table to read for each of the question, Key Code Extract:
    - df2 = sqlContext.read.parquet("/user/final_ontime.parquet")
    - df2.registerTempTable("ontime2")
    - df3 = sqlContext.read.load('hdfs:///user/L_AIRLINE_ID.csv', format='com.databricks.spark.csv', header='true', inferSchema='true',treatEmptyValuesAsNulls= 'true',nullValue="")
    - df3.registerTempTable("airline_lkp")

  - Rank the top 10 most popular airports by numbers of flights to/from the airport – Key Code Extract : sqlContext.sql("select A.Dest Airport ,(A.x+B.y) TOTALNUMFLIGHTS from (SELECT Dest,count(*) x FROM ontime2 group by Dest) A INNER JOIN (SELECT Origin,count(*) y FROM ontime2 group by Origin) B ON A.Dest = B.Origin order by TOTALNUMFLIGHTS desc").show()
  - Rank the top 10 airlines by on-time arrival performance. - Key Code Extract: sqlContext.sql("SELECT B.Description , avg(ArrDelay) ArrivalDelay FROM ontime2 A,airline_lkp B where A.AirlineID=B.code group by B.Description order by avg(ArrDelay) asc LIMIT 10 ").show()
  - Rank the days of the week by on-time arrival performance. - Key Code Extract: sqlContext.sql("SELECT DayOfWeek , avg(ArrDelay) FROM ontime2 group by DayOfWeek").show()
  - For each airport X, rank the top-10 carriers in decreasing order of on-time departure performance from X. – Created a Cassandra table for the question and used spark sql query to have answer data set. cleaned all the nulls before loading the data into Cassandra table Key Code Extract:
    - CREATE TABLE QUES2I (Airport varchar,   Carrier varchar,   DepartureDelay float, PRIMARY KEY ((Airport),DepartureDelay,Carrier));
    - res2I = sqlContext.sql("SELECT Origin airport ,UniqueCarrier carrier ,avg(DepDelay) departuredelay FROM ontime2 group by Origin,UniqueCarrier ")
    - res21= res2I.na.fill(0).show()
    - res2I.write\
    -     .format("org.apache.spark.sql.cassandra")\
    -     .mode('append')\
    -     .options(table="ques2i", keyspace="demodb")\
    -     .save()
    -
  - For each airport X, rank the top-10 airports in decreasing order of on-time departure performance from X. Created a Cassandra table for the question and used spark sql query to have answer data set. cleaned all the nulls before loading the data into Cassandra table, Key Code Extract:

- CREATE TABLE QUES2II (org_airport varchar,   dest_airport varchar,   DepartureDelay float,   PRIMARY KEY ((org_airport),DepartureDelay,dest_airport));
- res2II = sqlContext.sql("SELECT Origin org_airport ,Dest dest_airport ,avg(DepDelay) departuredelay FROM ontime2 group by Origin,Dest ")
- res22= res2II.na.fill(0)
- res22.write\
-     .format("org.apache.spark.sql.cassandra")\
-     .mode('append')\
-     .options(table="ques2ii", keyspace="demodb")\
-     .save()

- For each source-destination pair X-Y, rank the top-10 carriers in decreasing order of on-time arrival performance at Y from X- Key Code Extract: Created a Cassandra table for the question and used spark sql query to have answer data set. cleaned all the nulls before loading the data into Cassandra table, Key Code Extract:
    - CREATE TABLE QUES2III (org_airport varchar,   dest_airport varchar, Carrier varchar, ArrivalDelay float,   PRIMARY KEY ((org_airport,dest_airport),ArrivalDelay,Carrier));
    -
    - res2III = sqlContext.sql("SELECT Origin org_airport ,Dest dest_airport,UniqueCarrier carrier ,avg(ArrDelay) arrivaldelay FROM ontime2 group by Origin,Dest,UniqueCarrier ")
    - res23= res2III.na.fill(0)
    - res23.write\
    -     .format("org.apache.spark.sql.cassandra")\
    -     .mode('append')\
    -     .options(table="ques2iii", keyspace="demodb")\
    -     .save()
- For each source-destination pair X-Y, determine the mean arrival delay (in minutes) for a flight from X to Y. Created a Cassandra table for the question and used spark sql query to have answer data set. cleaned all the nulls before loading the data into Cassandra table, Key Code Extract:
    - CREATE TABLE QUES2IV (org_airport varchar,   dest_airport varchar, ArrivalDelay float,   PRIMARY KEY ((org_airport,dest_airport),ArrivalDelay));
    -
    - res2IV = sqlContext.sql("SELECT Origin org_airport ,Dest dest_airport,avg(ArrDelay) arrivaldelay FROM ontime2 group by Origin,Dest ")
    - res24= res2IV.na.fill(0)
    - res24.write\
    -     .format("org.apache.spark.sql.cassandra")\
    -     .mode('append')\
    -     .options(table="ques2iv", keyspace="demodb")\
    -     .save()

- Tom wants to travel from airport X to airport Z. However, Tom also wants to stop at airport Y for some sightseeing on the way. More concretely, Tom has the following requirements (for specific queries, see the **Task 1 Queries** and **Task 2 Queries**):
    - a) The second leg of the journey (flight Y-Z) must depart two days after the first leg (flight X-Y). For example, if X-Y departs on January 5, 2008, Y-Z must depart on January 7, 2008.
    - b) Tom wants his flights scheduled to depart airport X *before* 12:00 PM local time and to depart airport Y *after* 12:00 PM local time.
    - c) Tom wants to arrive at each destination with as little delay as possible. You can assume you know the actual delay of each flight. Created a Cassandra table for the question and used spark sql query to have answer data set. cleaned all the nulls before loading the data into Cassandra table, Key Code Extract:
        - res3II = sqlContext.sql("SELECT A.origin st_airport,A.dest intrm_airport,A.AirlineID st_airline,A.FlightNum st_flt,A.FlightDate st_flt_dt,A.ArrDelay st_dly ,A.CRSDepTime st_dep_tm ,B.origin conn_airport,B.dest conn_dst,B.AirlineID conn_arln,B.FlightNum conn_flt,B.FlightDate conn_flt_dt,B.ArrDelay conn_dely,B.CRSDepTime conn_schd_dep,COALESCE((A.ArrDelay+B.ArrDelay),0) tot_delay FROM ontime2 A,ontime2 B \
        - where A.dest=B.origin and B.FlightDate = date_add(A.FlightDate,2) and \
        - A.CRSDepTime < 1200 and B.CRSDepTime > 1200 ")
        - res32= res3II.na.fill(0)
        - res32.write\
        - .format("org.apache.spark.sql.cassandra")\
        - .mode('append')\
        - .options(table="ques3ii", keyspace="demodb")\
        - .save()

- What are the results of each question? Use only the provided subset for questions from Group 2 and Question 3.2.
    - 1.1 Rank the top 10 most popular airports by numbers of flights to/from the airport.

```
+-------+---------------+

|Airport|TOTALNUMFLIGHTS|

+-------+---------------+

|    ORD|       12449354|
```

```
|   ATL|     11540422|
|   DFW|     10799303|
|   LAX|      7723596|
|   PHX|      6585534|
|   DEN|      6273787|
|   DTW|      5636622|
|   IAH|      5480734|
|   MSP|      5199213|
|   SFO|      5171023|
|   EWR|      5136971|
|   STL|      5125336|
|   LAS|      4962958|
|   CLT|      4824711|
|   LGA|      4337167|
|   BOS|      4311116|
|   PHL|      4079651|
|   PIT|      3936220|
|   SLC|      3815114|
|   SEA|      3736761|
+-------+---------------+
only showing top 20 rows
```

- 1.2 Rank the top 10 airlines by on-time arrival performance.

```
+--------------------+------------------+
|         Description|      ArrivalDelay|
```

```
+--------------------+------------------+

|Hawaiian Airlines...| -1.01180434574519|

|Aloha Airlines In...|1.1569234424812056|

|Pacific Southwest...|1.4506385127822803|

|Midway Airlines I...| 4.747609195734892|

|Pan American Worl...|5.3224309999287875|

|Frontier Airlines...| 5.465881148819851|

|Northwest Airline...| 5.557783392671835|

|Southwest Airline...|5.5607742598815735|

|SkyWest Airlines ...| 5.736312463662878|

|Endeavor Air Inc....|5.8671846616957595|

+--------------------+------------------+
```

- 1.3 Rank the days of the week by on-time arrival performance.

```
+---------+-----------------------------+

|DayOfWeek|avg(CAST(ArrDelay AS DOUBLE))|

+---------+-----------------------------+

|        7|            6.613280292442754|

|        3|            7.203656394670348|

|        5|            9.721032337585571|

|        6|            4.301669926076596|

|        1|            6.716102802585582|

|        4|            9.094441008336657|

|        2|            5.990458841319885|

+---------+-----------------------------+
```

- 2.1 For each airport X, rank the top-10 carriers in decreasing order of on-time departure performance from X.

```
select airport,carrier,departuredelay from demodb.ques2i where airport= 'CMI';
```

```
airport | carrier | departuredelay

---------+---------+----------------

    CMI |      OH |       0.611626

    CMI |      US |        2.03305

    CMI |      TW |        4.12062

    CMI |      PI |       4.45563

    CMI |      DH |        6.02789

    CMI |      EV |        6.66514

    CMI |      MQ |          8.016
```

```
select airport,carrier,departuredelay from demodb.ques2i where airport= 'BWI' limit 10;
```

```
airport | carrier | departuredelay

---------+---------+----------------

    BWI |      F9 |       0.756244

    BWI |  PA (1) |         4.7619

    BWI |      CO |        5.17934

    BWI |      YV |         5.4965

    BWI |      NW |        5.70557

    BWI |      AA |        6.00285

    BWI |      9E |        7.23981

    BWI |      US |         7.4944

    BWI |      DL |        7.67682
```

```
   BWI |     UA |       7.73792
```

```
select airport,carrier,departuredelay from demodb.ques2i where airport= 'MIA' limit 10
;
```

```
airport | carrier | departuredelay
---------+---------+----------------
   MIA |     9E |           -3
   MIA |     EV |       1.20264
   MIA |     TZ |       1.78224
   MIA |     XE |       1.87319
   MIA |  PA (1) |          4.2
   MIA |     NW |       4.50167
   MIA |     US |       6.09067
   MIA |     UA |       6.86973
   MIA |  ML (1) |       7.50455
   MIA |     FL |       8.56511
```

```
select airport,carrier,departuredelay from demodb.ques2i where airport= 'LAX' limit 10;
```

```
airport | carrier | departuredelay
---------+---------+----------------
   LAX |     MQ |       2.40722
   LAX |     OO |       4.22196
   LAX |     FL |       4.72513
   LAX |     TZ |       4.76394
   LAX |     PS |       4.86034
   LAX |     NW |       5.11955
```

```
    LAX |      F9 |       5.72916
    LAX |      HA |       5.81365
    LAX |      YV |       6.02416
    LAX |      US |       6.7464
```

select airport,carrier,departuredelay from demodb.ques2i where airport= 'IAH' limit 10;

```
airport | carrier | departuredelay
---------+---------+----------------
    IAH |      NW |       3.56371
    IAH |  PA (1) |       3.98473
    IAH |      PI |       3.98867
    IAH |      US |       5.06027
    IAH |      F9 |       5.54524
    IAH |      AA |       5.70396
    IAH |      TW |       6.04878
    IAH |      WN |       6.23113
    IAH |      OO |       6.58796
    IAH |      MQ |       6.71297
```

select airport,carrier,departuredelay from demodb.ques2i where airport= 'SFO' limit 10;

```
airport | carrier | departuredelay
---------+---------+----------------
    SFO |      TZ |       3.95242
    SFO |      MQ |       4.85392
    SFO |      F9 |       5.16244
    SFO |  PA (1) |       5.28761
```

```
SFO |     NW |       5.75781
SFO |     PS |       6.30352
SFO |     DL |       6.56273
SFO |     CO |       7.08305
SFO |     US |       7.52751
SFO |     TW |       7.79488
```

- 2.2 For each airport X, rank the top-10 airports in decreasing order of on-time departure performance from X.

```
select org_airport,dest_airport,departuredelay from demodb.ques2i where org_airport = 'CMI' limit 10;
```

```
org_airport | dest_airport | departuredelay
-------------+--------------+----------------
       CMI |        ABI |           -7
       CMI |        PIT |       1.10243
       CMI |        CVG |       1.89476
       CMI |        DAY |       3.11624
       CMI |        STL |       3.98167
       CMI |        PIA |       4.59189
       CMI |        DFW |       5.94414
       CMI |        ATL |       6.66514
       CMI |        ORD |       8.1941
```

```
select org_airport,dest_airport,departuredelay from demodb.ques2ii where org_airport = 'BWI' limit 10;
```

```
org_airport | dest_airport | departuredelay
-------------+--------------+----------------
```

```
       BWI |        SAV |          -7
       BWI |        MLB |       1.15537
       BWI |        DAB |       1.46959
       BWI |        SRQ |       1.58848
       BWI |        IAD |       1.79094
       BWI |        UCA |       3.65417
       BWI |        CHO |       3.74493
       BWI |        GSP |       4.19769
       BWI |        SJU |       4.44466
       BWI |        OAJ |       4.47111
```

```
select org_airport,dest_airport,departuredelay from demodb.ques2ii where org_airport = 'MIA' limit 10;
```

```
org_airport | dest_airport | departuredelay
-------------+--------------+----------------
       MIA |        SHV |           0
       MIA |        BUF |           1
       MIA |        SAN |       1.71038
       MIA |        SLC |       2.53719
       MIA |        HOU |        2.9122
       MIA |        ISP |        3.6474
       MIA |        MEM |       3.74511
       MIA |        PSE |       3.97585
       MIA |        TLH |       4.26148
       MIA |        MCI |       4.61225
```

```
select org_airport,dest_airport,departuredelay from demodb.ques2ii where org_airport = 'LAX' limit 10;
```

```
org_airport | dest_airport | departuredelay
-------------+--------------+----------------
        LAX |         SDF |          -16
        LAX |         IDA |           -7
        LAX |         DRO |           -6
        LAX |         RSW |           -3
        LAX |         LAX |           -2
        LAX |         BZN |    -0.727273
        LAX |         MAF |            0
        LAX |         PIH |            0
        LAX |         IYK |      1.26982
        LAX |         MFE |      1.37647
```

```
select org_airport,dest_airport,departuredelay from demodb.ques2ii where org_airport = 'IAH' limit 10;
```

```
org_airport | dest_airport | departuredelay
-------------+--------------+----------------
        IAH |         MSN |           -2
        IAH |         AGS |    -0.618791
        IAH |         MLI |         -0.5
        IAH |         EFD |      1.88771
        IAH |         HOU |      2.17204
        IAH |         JAC |      2.57059
```

```
       IAH |         MTJ |       2.95016

       IAH |         RNO |       3.22158

       IAH |         BPT |       3.59953

       IAH |         VCT |       3.61191
```

```
select org_airport,dest_airport,departuredelay from demodb.ques2ii where org_airport = 'SFO' limit 10;
```

```
org_airport | dest_airport | departuredelay
-------------+--------------+----------------
       SFO |         SDF |          -10
       SFO |         MSO |           -4
       SFO |         PIH |           -3
       SFO |         LGA |      -1.75758
       SFO |         PIE |      -1.34104
       SFO |         OAK |       -0.8132
       SFO |         FAR |            0
       SFO |         BNA |       2.42597
       SFO |         MEM |       3.30248
       SFO |         SCK |            4
```

- 2.3 For each source-destination pair X-Y, rank the top-10 carriers in decreasing order of on-time arrival performance at Y from X-

```
select org_airport,dest_airport,carrier,arrivaldelay from demodb.ques2iii where org_airport = 'CMI' and dest_airport= 'ORD' limit 10;

org_airport | dest_airport | carrier | arrivaldelay
-------------+--------------+---------+--------------
       CMI |         ORD |      MQ |     10.14366
```

```
select org_airport,dest_airport,carrier,arrivaldelay from demodb.ques2iii where
org_airport = 'IND' and dest_airport= 'CMH' limit 10;

org_airport | dest_airport | carrier | arrivaldelay

-------------+--------------+---------+--------------

        IND |          CMH |      CO |     -2.54585

        IND |          CMH |      AA |          5.5

        IND |          CMH |      HP |      5.69726

        IND |          CMH |      NW |      5.76154

        IND |          CMH |      US |      6.87847

        IND |          CMH |      DL |      10.6875

        IND |          CMH |      EA |     10.81308

select org_airport,dest_airport,carrier,arrivaldelay from demodb.ques2iii where
org_airport = 'DFW' and dest_airport= 'IAH' limit 10;

org_airport | dest_airport | carrier | arrivaldelay

-------------+--------------+---------+--------------

        DFW |          IAH |  PA (1) |     -1.59649

        DFW |          IAH |      EV |      5.09251

        DFW |          IAH |      UA |       5.4142

        DFW |          IAH |      CO |      6.49373

        DFW |          IAH |      OO |      7.56401

        DFW |          IAH |      XE |      8.09429

        DFW |          IAH |      AA |      8.38123

        DFW |          IAH |      DL |      8.59851

        DFW |          IAH |      MQ |      9.10321
```

```
select org_airport,dest_airport,carrier,arrivaldelay from demodb.ques2iii where
org_airport = 'LAX' and dest_airport= 'SFO' limit 10;

org_airport | dest_airport | carrier | arrivaldelay

-------------+--------------+---------+--------------

       LAX |         SFO |      TZ |     -7.61905

       LAX |         SFO |      PS |     -2.14634

       LAX |         SFO |      F9 |     -2.02869

       LAX |         SFO |      EV |      6.96463

       LAX |         SFO |      AA |      7.38679

       LAX |         SFO |      MQ |      7.80776

       LAX |         SFO |      US |      7.96472

       LAX |         SFO |      WN |      8.79205

       LAX |         SFO |      CO |      9.35478

       LAX |         SFO |      NW |      9.84879

select org_airport,dest_airport,carrier,arrivaldelay from demodb.ques2iii where
org_airport = 'JFK' and dest_airport= 'LAX' limit 10;

org_airport | dest_airport | carrier | arrivaldelay

-------------+--------------+---------+--------------

       JFK |         LAX |      B6 |            0

       JFK |         LAX |      UA |      3.31387

       JFK |         LAX |      HP |       6.6806

       JFK |         LAX |      AA |      6.90372

       JFK |         LAX |      DL |      7.93446

       JFK |         LAX |  PA (1) |     11.01944

       JFK |         LAX |      TW |     11.70201
```

```
select org_airport,dest_airport,carrier,arrivaldelay from demodb.ques2iii where org_airport = 'ATL' and dest_airport= 'PHX' limit 10;

org_airport | dest_airport | carrier | arrivaldelay
-------------+--------------+---------+--------------
        ATL |          PHX |      FL |      4.55263
        ATL |          PHX |      US |      6.28812
        ATL |          PHX |      HP |      8.48144
        ATL |          PHX |      EA |      8.95357
        ATL |          PHX |      DL |      9.80828
```

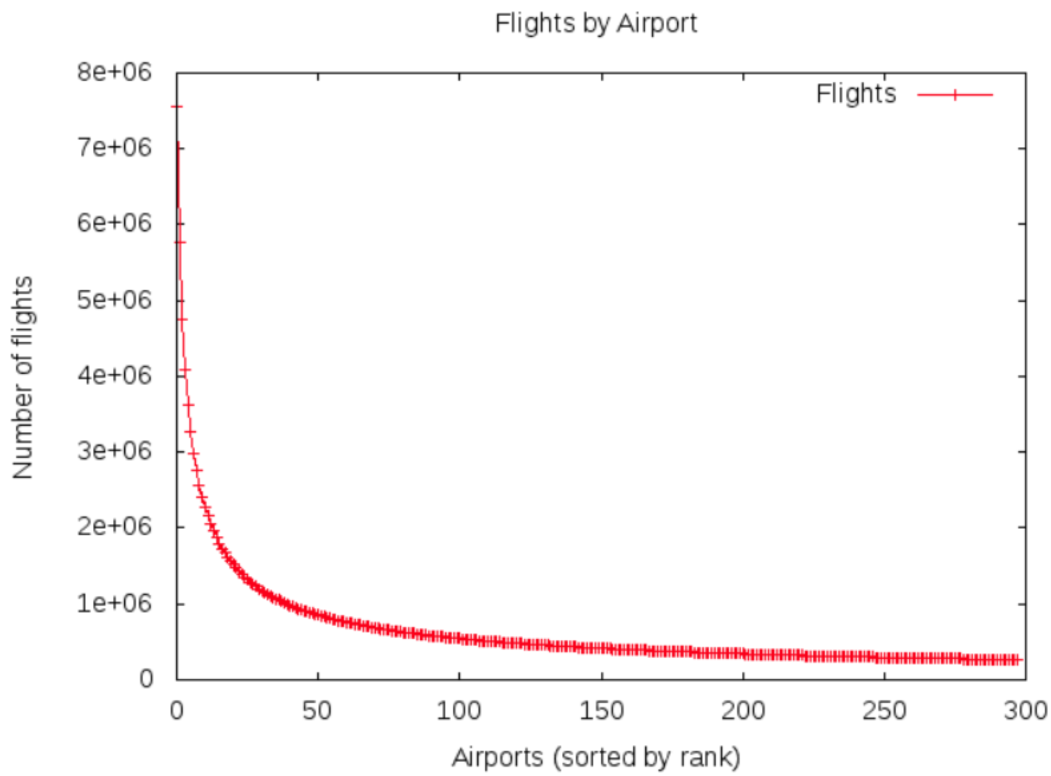
- 2.4 For each source-destination pair X-Y, determine the mean arrival delay (in minutes) for a flight from X to Y.

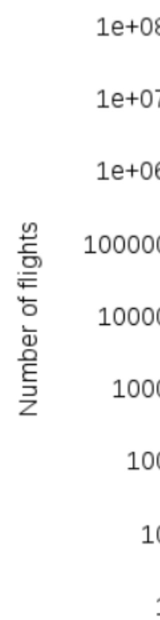
```
select org_airport,dest_airport,arrivaldelay from demodb.ques2iv where org_airport = 'CMI' and dest_airport= 'ORD' limit 10;

org_airport | dest_airport | arrivaldelay
-------------+--------------+--------------
        CMI |          ORD |     10.14366

select org_airport,dest_airport,arrivaldelay from demodb.ques2iv where org_airport = 'IND' and dest_airport= 'CMH' limit 10;

org_airport | dest_airport | arrivaldelay
-------------+--------------+--------------
        IND |          CMH |       2.8999

select org_airport,dest_airport,arrivaldelay from demodb.ques2iv where org_airport = 'DFW' and dest_airport= 'IAH' limit 10;

org_airport | dest_airport | arrivaldelay
```

```
-------------+--------------+--------------
     DFW |        IAH |     7.65444

select org_airport,dest_airport,arrivaldelay from demodb.ques2iv where org_airport = 'LAX' and dest_airport= 'SFO' limit 10;

org_airport | dest_airport | arrivaldelay

-------------+--------------+--------------
     LAX |        SFO |     9.58928

select org_airport,dest_airport,arrivaldelay from demodb.ques2iv where org_airport = 'JFK' and dest_airport= 'LAX' limit 10;

org_airport | dest_airport | arrivaldelay

-------------+--------------+--------------
     JFK |        LAX |     6.63512

select org_airport,dest_airport,arrivaldelay from demodb.ques2iv where org_airport = 'ATL' and dest_airport= 'PHX' limit 10;

org_airport | dest_airport | arrivaldelay

-------------+--------------+--------------
     ATL |        PHX |     9.02134
```

Question 3.1 is all about analysis of data report. Zipf's law stats that given some corpus or natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table. In our case it means that the airport with a higher rank should have a double number of flights compare with the next airport in rank.Even if from the Flights by Airport figure we can hope for a Zipf distribution, after doing a log-log plot we can see that Airports rank by number of flights is not following this distribution (log log for Zipf should be a straight line and is not). Using special statistical tools (R, Python) it can be prove that this distribution is not Zipf but more a

Lognormal one since the bottom half look very different than the top half.





3.2 Tom wants to travel from airport X to airport Z. However, Tom also wants to stop at airport Y for some sightseeing on the way. More concretely, Tom has the following requirements (for specific queries, see the **Task 1 Queries** and **Task 2 Queries**):

a) The second leg of the journey (flight Y-Z) must depart two days after the first leg (flight X-Y). For example, if X-Y departs on January 5, 2008, Y-Z must depart on January 7, 2008.

b) Tom wants his flights scheduled to depart airport X *before* 12:00 PM local time and to depart airport Y *after* 12:00 PM local time.

c) Tom wants to arrive at each destination with as little delay as possible. You can assume you know the actual delay of each flight.

CMI → ORD → LAX, 04/03/2008

select * from demodb.ques3ii where st_airport= 'CMI' and intrm_airport= 'ORD' and conn_dst = 'LAX' and st_flt_dt ='2008-03-04' limit 1;

st_airport | intrm_airport | conn_dst | st_flt_dt | tot_delay | conn_airport | conn_arln | conn_dely | conn_flt | conn_flt_dt | conn_schd_dep | st_airline | st_dep_tm | st_dly | st_flt

```
------------+---------------+----------+------------+-----------+--------------+-----------+-----------+--------
--+-------------+---------------+------------+-----------+--------+--------

     CMI |         ORD |     LAX | 2008-03-04 |      -38 |         ORD |    19805 |      -24 |
607 |  2008-03-06 |        1950 |     20398 |     0710 |    -14 |   4278
```

JAX → DFW → CRP, 09/09/2008

```
select * from demodb.ques3ii where st_airport= 'JAX' and intrm_airport= 'DFW' and
conn_dst = 'CRP' and st_flt_dt ='2008-09-09' limit 1;

st_airport | intrm_airport | conn_dst | st_flt_dt  | tot_delay | conn_airport | conn_arln
| conn_dely | conn_flt | conn_flt_dt | conn_schd_dep | st_airline | st_dep_tm | st_dly
| st_flt

------------+---------------+----------+------------+-----------+--------------+-----------+-----------+--------
--+-------------+---------------+------------+-----------+--------+--------

     JAX |         DFW |     CRP | 2008-09-09 |       -6 |         DFW |    20398 |       -7 |
3627 |  2008-09-11 |        1645 |     19805 |     0725 |      1 |    845
```

SLC → BFL → LAX, 01/04/2008

```
select * from demodb.ques3ii where st_airport= 'SLC' and intrm_airport= 'BFL' and
conn_dst = 'LAX' and st_flt_dt ='2008-04-01' limit 1;

st_airport | intrm_airport | conn_dst | st_flt_dt  | tot_delay | conn_airport | conn_arln
| conn_dely | conn_flt | conn_flt_dt | conn_schd_dep | st_airline | st_dep_tm | st_dly
| st_flt

------------+---------------+----------+------------+-----------+--------------+-----------+-----------+--------
--+-------------+---------------+------------+-----------+--------+--------

     SLC |         BFL |     LAX | 2008-04-01 |       18 |         BFL |    20304 |        6 |
5429 |  2008-04-03 |        1455 |     20304 |     1100 |     12 |   3755
```

LAX → SFO → PHX, 12/07/2008

```
select * from demodb.ques3ii where st_airport= 'LAX' and intrm_airport= 'SFO' and
conn_dst = 'PHX' and st_flt_dt ='2008-07-12' limit 1;

st_airport | intrm_airport | conn_dst | st_flt_dt  | tot_delay | conn_airport | conn_arln
| conn_dely | conn_flt | conn_flt_dt | conn_schd_dep | st_airline | st_dep_tm | st_dly
| st_flt
```

```
------------+---------------+----------+------------+-----------+--------------+-----------+-----------+----------+-------------+---------------+------------+-----------+--------+--------

      LAX |         SFO |     PHX | 2008-07-12 |      -32 |        SFO |    20355 |      -19 |
412 |  2008-07-14 |        1925 |     19393 |     0650 |    -13 |   3534
```

DFW → ORD → DFW, 10/06/2008

```
select * from demodb.ques3ii where st_airport= 'DFW' and intrm_airport= 'ORD' and
conn_dst = 'DFW' and st_flt_dt ='2008-06-10' limit 1;

st_airport | intrm_airport | conn_dst | st_flt_dt  | tot_delay | conn_airport | conn_arln
| conn_dely | conn_flt | conn_flt_dt | conn_schd_dep | st_airline | st_dep_tm | st_dly
| st_flt

------------+---------------+----------+------------+-----------+--------------+-----------+-----------+----------+-------------+---------------+------------+-----------+--------+--------

      DFW |         ORD |     DFW | 2008-06-10 |      -31 |        ORD |    19805 |      -10 |
2341 |  2008-06-12 |        1645 |     19977 |     0700 |    -21 |   1104
```

LAX → ORD → JFK, 01/01/2008

```
select * from demodb.ques3ii where st_airport= 'LAX' and intrm_airport= 'ORD' and
conn_dst = 'JFK' and st_flt_dt ='2008-01-01' limit 1;

st_airport | intrm_airport | conn_dst | st_flt_dt  | tot_delay | conn_airport | conn_arln
| conn_dely | conn_flt | conn_flt_dt | conn_schd_dep | st_airline | st_dep_tm | st_dly
| st_flt

------------+---------------+----------+------------+-----------+--------------+-----------+-----------+----------+-------------+---------------+------------+-----------+--------+--------

      LAX |         ORD |     JFK | 2008-01-01 |       -6 |        ORD |    20409 |       -7 |
918 |  2008-01-03 |        1900 |     19977 |     0705 |      1 |    944
```

- What system- or application-level optimizations (if any) did you employ?
  - Data Ingestion & Acquisition – Used Spark program to read multiple csv files from HDFS to consolidate and convert into a more optimal format (Parquet) from storage and further processing perspective.
  - Storage – HDFS - Chose to convert csv data and store the data set as parquet columnar storage layout such as Parquet can speed up queries because it examines and performs calculations on all values for required columns only thereby reading only a small fraction of the data from a

data file or table. Parquet also supports flexible compression options so on-disk storage can be reduced drastically.

  - Storage savings - That is close to a 75% saving on storage for 1TB worth of data files the following Linux output shows on-HDFS size comparison between CSV and PARQUET: hadoop fs -du -h -s /user

- Data Processing - Spark
  - Used SparkSQL – While Spark accepts SQL the framework will translate commands into code that is processed by Executors. Below are the tuning options considered
    - File Formats- Processing query performance boost can reach 30X or higher in some cases in case of reading the data from parquet format file as compared to csv based. Leveraged the spark-sql-perf test kit to do query testing . Paruet helps to achieve less I/0
    - Join Optimization - Reduce resource consumption during the Spark shuffle stage of execution by sending out data from a smaller table (like Look up tables) in the join through a Broadcast Join configured with spark.sql.autoBroadcastJoinThreshold
    - Shuffle Partitions- SparkSQL requires the use of partitions to perform many of the tasks that are submitted via SQL such as aggregations, groupings, joins and filtering. The number of partitions involved in the shuffle - and thus the measure of parallelism - is determined by spark.sql.shuffle.partitions.
  - Cluster Resource Tuning Example
    - Dynamic allocation allows Spark to dynamically scale the cluster resources allocated to your application based on the workload. When dynamic allocation is enabled and a Spark application has a backlog of pending tasks, it can request executors. When the application becomes idle, its executors are released and can be acquired by other applications.Enable it by setting proposert spark.dynamicAllocation.enabled ='True'
  - Tuning the Number of Partitions
    - Used Parquet to create more splits.
  - Network optimization - Ran the Spark application in client mode to submit my application from a gateway machine(master node/Yarn Resource manager) that is physically co-located with the worker machines for minimizing network latency between the drivers and the executors
- Read Performance -DB model Cassandra
  - Dedicated Commit Log Disk: Cassandra write operations are occurred on a commit log on disk and then to an in-memory table structure called Memtable. When thresholds are reached, that Memtable is flushed to a disk in a format called SSTable. So if you separate out Commit Log locations, it will isolate Commit Log I/O traffics from other Cassandra Reads, Memtables and SSTables traffics
    - Mount a separate partition for commit log
    - Changed CommitLogDirectory: /mnt/commitlog in cassandra.yaml
  - Increasing Java Heap Size: Cassandra runs on JVM. So you might face out of memory issues when you run a heavy load on Cassandra.Followed following rule and updated the heap size as 2 GB in cassandra-env.sh f
    - Heap Size = 1/2 of System Memory when System Memory < 2GB
    - Heap Size = 1GB when System Memory >= 2GB and <= 4GB

        - Heap Size = 1/4 of System Memory(but not more than 8GB) when System Memory >4GB
    - Tune Concurrent Reads and Writes: Concurrent readers and writers control the maximum number of threads allocated to a particular stage. So having an optimal concurrent reads and concurrent writes value will improve Cassandra performance .Changed two parameters ConcurrentReaders and ConcurrentWriters in cassandra.yaml by folliwng the rule 4 concureent reads per processor core so for t2.large it will be 16

- Give your opinion about whether the results make sense and are useful in any way.

Results are useful as it gives insights on the flights data in terms of popularity of airport and could help coming up with the optimized itinerary with different constraints and conditions based on the past 20-year data.

Further data analysis will help in understanding the problematic airports or carrier having arrival and departure delay issues for root cause analysis and subsequent corrective and preventive actions.