Jaskeerat Singh

EMP ID: 8358

ZOMATO PROJECT

**Q1**

Download the zomato.zip file and put it on the home

**Q2**

Put these json files onto the hdfs

**Make a folder structure on hdfs:**

hdfs dfs -mkdir zomato\_etl/source

hdfs dfs -mkdir zomato\_etl/source/json

hdfs dfs -mkdir zomato\_etl/source/csv

**Put the json files on hdfs in zomato\_etl/source/json:**

hdfs dfs -put file1.json zomato\_etl/source/json

hdfs dfs -put file2.json zomato\_etl/source/json

hdfs dfs -put file3.json zomato\_etl/source/json

**SPARK CODE for converting Json to CSV**

import java.text.SimpleDateFormat

import com.typesafe.config.{Config, ConfigFactory}

import java.io.{File, InputStream}

import java.net.URLDecoder

import java.security.MessageDigest

import java.util.Calendar

import org.apache.commons.net.util.SubnetUtils

import org.apache.hadoop.fs.{FileSystem,Path}

import org.apache.spark.graphx.{EdgeDirection, \_}

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.{StructField, StructType}

import java.io.PrintWriter

import org.apache.spark.sql.functions.explode

import org.apache.spark.sql.SparkSession

import org.apache.spark.sql.functions.col

import org.apache.spark.sql.SQLImplicits

object jsoncsv {

def main(args: Array[String]) = {

val spark = SparkSession

.builder()

.appName("jsoncsv")

.config("spark.sql.warehouse.dir","/warehouse/tablespace/managed/hive")

.config("hive.metastore.uris", "thrift://hdp2.infocepts.com:9083")

.master("local")

.enableHiveSupport()

.getOrCreate()

var a = args(0)

var b = args(1)

val jsonfile = "/user/bctfeb20-bigdata-jaskeerat.s/"+a

val csvfile = "zomato\_etl/source/"+b

val rdd = spark.read.format("json").load(jsonfile)

val rdd1 = rdd.select(explode(col("restaurants.restaurant")))

val rdd2 = rdd1.select(col("col.R.res\_id") as "Restaurant Id",col("col.name") as "Restaurant Name",col("col.location.country\_id") as "Country Code",col("col.location.city") as "City",col("col.location.address") as "Address",col("col.location.locality") as "Locality",col("col.location.locality\_verbose") as "Locality Verbose",col("col.location.longitude") as "Longitude",col("col.location.latitude") as "Latitude",col("col.cuisines") as "Cuisines",col("col.average\_cost\_for\_two") as "Average Cost for two",col("col.currency") as "Currency",col("col.has\_table\_booking") as "Has Table booking",col("col.has\_online\_delivery") as "Has Online delivery",col("col.is\_delivering\_now") as "Is delivering now",col("col.switch\_to\_order\_menu") as "Switch to order menu",col("col.price\_range") as "Price range", col("col.user\_rating.aggregate\_rating") as "Aggregate rating",col("col.user\_rating.rating\_text") as "Rating text",col("col.user\_rating.votes") as "Votes")

rdd2.write.option("header","true").csv(csvfile)

spark.stop()

}

}

SHELL SCRIPT to run the spark application

To run the shell script

Make a empty file: **touch shell.sh**

Open the file: **vi shell.sh**

Write the code:

#!bin/bash

echo "Enter json file name"

read c

echo "Enter json file name"

read d

spark-submit --master yarn --deploy-mode "client" --name "testingApplication" --driver-memory 512m --driver-cores 1 --num-executors 3 --executor-cores 1 --executor-memory 512m --class jsoncsv /home/bctfeb20-bigdata-jaskeerat.s/test\_project\_jaskeerat/target/scala-2.11/test\_project\_jaskeerat\_2.11-1.0.jar $c $d

Make the file executable: **Chmod u+x shell.sh**

Execute the shell script: **source shell.sh**

**Q3**

**Move the CSV files to the hdfs in zomato\_etl/source/csv folder**

hdfs dfs -put zomato\_20190609.csv zomato\_etl/source/csv

hdfs dfs -put zomato\_20190610.csv zomato\_etl/source/csv

hdfs dfs -put zomato\_20190611.csv zomato\_etl/source/csv

hdfs dfs -put zomato\_20190612.csv zomato\_etl/source/csv

hdfs dfs -put zomato\_20190613.csv zomato\_etl/source/csv

**Creating the zomato\_jassi\_source table:**

CREATE EXTERNAL TABLE

bct\_bi\_20.zomato\_jassi\_source(

`Restaurant ID` INT,

`Restaurant Name` STRING,

`Country Code` INT,

`City` STRING,

`Address` STRING,

`Locality` STRING,

`Locality Verbose` STRING,

`Longitude` STRING,

`Latitude` STRING,

`Cuisines` STRING,

`Average Cost for two` INT,

`Currency` STRING,

`Has Table booking` INT,

`Has Online delivery` INT,

`Is delivering now` INT,

`Switch to order menu` INT,

`Price range` INT,

`Aggregate rating` STRING,

`Rating text` STRING,

`Votes` INT

)

PARTITIONED BY (`filedate` INT)

ROW FORMAT SERDE

'org.apache.hadoop.hive.serde2.OpenCSVSerde'

WITH SERDEPROPERTIES (

'escapeChar'='\\',

'quoteChar'='"',

'separatorChar'=',')

stored as textfile

location '/user/bctfeb20-bigdata-jaskeerat.s/zomato\_etl/zomatojassiexample3';

**LOAD THE CSV FILES INTO THE TABLE**

load data inpath /user/bctfeb20-bigdata-jaskeerat.s/zomato\_etl/source/csv/zomato\_20190609.csv/part-00000-4d01a7b2-771e-4a72-a3c7-71f17ab80d7d-c000.csv' into table zomato\_jassi partition(filedate = 20190609);

load data inpath '/user/bctfeb20-bigdata-jaskeerat.s/zomato\_etl/source/csv/zomato\_20190610.csv//part-00000-18827d24-985f-4904-8385-84b8c6466acd-c000.csv' into table zomato\_jassi partition(filedate = 20190610)

load data inpath '/user/bctfeb20-bigdata-jaskeerat.s/zomato\_etl/source/csv/zomato\_20190611.csv/part-00000-7611ff95-dd35-4e67-be12-65d3837ad408-c000.csv' into table zomato\_jassi partition(filedate = 20190611);

load data inpath '/user/bctfeb20-bigdata-jaskeerat.s/zomato\_20190612.csv/part-00000-365d2327-1ed2-4961-9da6-9ea63750aef1-c000.csv' into table zomato\_jassi\_example partition(filedate = 20190612);

load data inpath '/user/bctfeb20-bigdata-jaskeerat.s/zomato\_20190613.csv/part-00000-986e3826-1ed2-4581-8fa6-94ad7750aef1-c000.csv' into table zomato\_jassi\_example partition(filedate = 20190613);

**CREATING TABLE dim\_country\_jassi**

create table bct\_bi\_20.dim\_country\_jassi(

`Country code` INT,

`Country` STRING)

ROW FORMAT DELIMITED

fields terminated by ','

stored as textfile;

**LOADING THE dim\_country\_jassi with country\_code.csv**

load data inpath '/user/bctfeb20-bigdata-jaskeerat.s/zomato\_etl/source/csv/country\_code.

csv' into table dim\_country\_jassi;

**CREATING TABLE zomato\_summary\_log\_jassi**

create table bct\_bi\_20.zomato\_summary\_log\_jassi(

`Job id` INT,

`Job step` STRING,

`spark submit command` STRING,

`Job start time` TIMESTAMP,

`Job end time` TIMESTAMP,

`Job status` INT)

ROW FORMAT DELIMITED

fields terminated by ','

stored as textfile;

**Q4**

**A)**

**CREATE A hivefile.hql file**

touch hivefile.hql

vi hivefile.hql

INSERT OVERWRITE table bct\_bi\_20.zomato\_summary\_jaskeerat\_managed

PARTITION(p\_filedate,p\_country\_name)

select

CAST(`Restaurant ID` as INT),

CAST(`Restaurant Name` as STRING),

CAST(zomato\_jassi\_source.`Country Code` as INT),

CAST(`City` as STRING),

CAST(`Address` as STRING),

CAST(`Locality` as STRING),

CAST(`Locality Verbose` as STRING),

CAST(`Longitude` as STRING),

CAST(`Latitude` as STRING),

CAST(`Cuisines` as STRING),

CAST(`Average Cost for two` as INT),

CAST(`Currency` as STRING),

CAST(`Has Table booking` as INT),

CAST(`Has Online delivery` as INT),

CAST(`Is delivering now` as INT),

CAST(`Switch to order menu` as INT),

CAST(`Price range` as INT),

CAST(`Aggregate rating` as DOUBLE),

CAST(`Rating text` as STRING),

CAST(`Votes` as INT),

null,

null,

FROM\_UNIXTIME( UNIX\_TIMESTAMP()),

CURRENT\_USER(),

CAST(zomato\_jassi\_source.`filedate` as INT),

CAST(dim\_country\_jassi.`Country` as STRING)

from bct\_bi\_20.zomato\_jassi\_source,bct\_bi\_20.dim\_country\_jassi

where zomato\_jassi\_source.`Country Code`=dim\_country\_jassi.`Country code`;

UPDATE bct\_bi\_20.zomato\_summary\_jaskeerat\_managed SET `m\_rating\_colour` = (

CASE WHEN (`rating text` = "Poor" and (`aggregate rating` >=1.9 and `aggregate rating` <=2.4)) THEN "Red"

WHEN (`rating text` = "Average" and (`aggregate rating` >=2.5 and `aggregate rating` <=3.4)) THEN "Amber"

WHEN (`rating text` = "Good" and (`aggregate rating` >=3.5 and `aggregate rating` <=3.9)) THEN "Light Green"

WHEN (`rating text` = "Very Good" and (`aggregate rating` >=4.0 and `aggregate rating` <=4.4)) THEN "Green"

WHEN (`rating text` = "Excellent" and (`aggregate rating` >=4.5 and `aggregate rating` <=5.0)) THEN "Gold"

ELSE 'NA' END);

UPDATE bct\_bi\_20.zomato\_summary\_jaskeerat\_managed SET `m\_cuisines` = (

CASE

WHEN (`cuisines` like '%Assamese%') THEN "Indian"

WHEN (`cuisines` like '%Andhra%') THEN "Indian"

WHEN (`cuisines` like '%Awadhi%') THEN "Indian"

WHEN (`cuisines` like '%Bengali%') THEN "Indian"

WHEN (`cuisines` like '%Bihari%') THEN "Indian"

WHEN (`cuisines` like '%Biryani%') THEN "Indian"

WHEN (`cuisines` like '%Chettinad%') THEN "Indian"

WHEN (`cuisines` like '%Curry%') THEN "Indian"

WHEN (`cuisines` like '%Gujrati%') THEN "Indian"

WHEN (`cuisines` like '%Goan%') THEN "Indian"

WHEN (`cuisines` like '%Hyderabadi%') THEN "Indian"

WHEN (`cuisines` like '%Indian%') THEN "Indian"

WHEN (`cuisines` like '%Kashmiri%') THEN "Indian"

WHEN (`cuisines` like '%Kerala%') THEN "Indian"

WHEN (`cuisines` like '%Lucknowi%') THEN "Indian"

WHEN (`cuisines` like '%Maharashtrian%') THEN "Indian"

WHEN (`cuisines` like '%Modern Indian%') THEN "Indian"

WHEN (`cuisines` like '%Mangalorean%') THEN "Indian"

WHEN (`cuisines` like '%Mughlai%') THEN "Indian"

WHEN (`cuisines` like '%Malwani%') THEN "Indian"

WHEN (`cuisines` like '%Mithai%') THEN "Indian"

WHEN (`cuisines` like '%Naga%') THEN "Indian"

WHEN (`cuisines` like '%North Indian%') THEN "Indian"

WHEN (`cuisines` like '%Oriya%') THEN "Indian"

WHEN (`cuisines` like '%Rajasthani%') THEN "Indian"

WHEN (`cuisines` like '%South Indian%') THEN "Indian"

WHEN (`cuisines` like '%Parsi%') THEN "Indian"

WHEN (`cuisines` = "North Indian") THEN "Indian"

WHEN (`cuisines` = "South Indian") THEN "Indian"

WHEN (`cuisines` = "Modern Indian") THEN "Indian"

ELSE 'World Cuisines' END);

UPDATE bct\_bi\_20.zomato\_summary\_jaskeerat\_managed SET `cuisines` = (

CASE

WHEN `cuisines` is null THEN "NA"

ELSE `cuisines` END);

UPDATE bct\_bi\_20.zomato\_summary\_jaskeerat\_managed SET `votes` = "NA" WHERE `votes` IS NULL;

UPDATE bct\_bi\_20.zomato\_summary\_jaskeerat\_managed SET `locality` = "NA" WHERE `locality` IS NULL;

UPDATE bct\_bi\_20.zomato\_summary\_jaskeerat\_managed SET `locality verbose` = "NA" WHERE `locality verbose` IS NULL;

UPDATE bct\_bi\_20.zomato\_summary\_jaskeerat\_managed SET `address` = "NA" WHERE `address` IS NULL;

UPDATE bct\_bi\_20.zomato\_summary\_jaskeerat\_managed SET `longitude` = "NA" WHERE `longitude` IS NULL;

UPDATE bct\_bi\_20.zomato\_summary\_jaskeerat\_managed SET `latitude` = "NA" WHERE `latitude` IS NULL;

UPDATE bct\_bi\_20.zomato\_summary\_jaskeerat\_managed SET `rating text` = "NA" WHERE `rating text` IS NULL;

**RUNNING THE SHELL SCRIPT FOR hivefile.hql FILE**

#!bin/bash

echo 'Running the hivefile.hql file'

hive -f hivefile.hql

echo 'all the transformations are successful'

**TO RUN THE hivefile.hql FILE**

Touch hivefile.sh

Vi hivefile.sh

Chmod u+x hivefile.sh

Source hivefile.sh

**B)**

import java.text.SimpleDateFormat

import com.typesafe.config.{Config, ConfigFactory}

import java.io.{File, InputStream}

import java.net.URLDecoder

import java.security.MessageDigest

import java.util.Calendar

import org.apache.commons.net.util.SubnetUtils

import org.apache.hadoop.fs.{FileSystem,Path}

import org.apache.spark.graphx.{EdgeDirection, \_}

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.{StructField, StructType}

import java.io.PrintWriter

import org.apache.spark.sql.functions.explode

import org.apache.spark.sql.SparkSession

import org.apache.spark.sql.functions.col

import org.apache.spark.sql.SQLImplicits

object jsoncsv {

def main(args: Array[String]) = {

val spark = SparkSession

.builder()

.appName("jsoncsv")

.config("spark.sql.warehouse.dir","/warehouse/tablespace/managed/hive")

.config("hive.metastore.uris", "thrift://hdp2.infocepts.com:9083")

.master("local")

.enableHiveSupport()

.getOrCreate()

var a = args(0)

var b = args(1)

var c = args(2)

spark.sql(s"load data inpath '$a' into Table $b partition(filedate = $c)")

spark.stop()

}

}

**MAKE A load.sh FILE**

Touch load.sh

Vi load.sh

Chmod u+x load.sh

#!bin/bash

echo "Enter the full path"

read a

echo "Enter the table name with database"

read b

echo "Enter the partition"

read c

spark-submit --master yarn --deploy-mode "client" --name "testingApplication" --driver-memory 512m --driver-cores 1 --num-executors 3 --executor-cores 1 --executor-memory 512m --class jsoncsv /home/bctfeb20-bigdata-jaskeerat.s/test\_project\_jaskeerat/target/scala-2.11/test\_project\_jaskeerat\_2.11-1.0.jar $a $b $c

Source load.sh

**Q5**

**MODULE 1**

import java.text.SimpleDateFormat

import com.typesafe.config.{Config, ConfigFactory}

import java.io.{File, InputStream}

import java.net.URLDecoder

import java.security.MessageDigest

import java.util.Calendar

import org.apache.commons.net.util.SubnetUtils

import org.apache.hadoop.fs.{FileSystem,Path}

import org.apache.spark.graphx.{EdgeDirection, \_}

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.{StructField, StructType}

import java.io.PrintWriter

import org.apache.spark.sql.functions.explode

import org.apache.spark.sql.SparkSession

import org.apache.spark.sql.functions.col

import org.apache.spark.sql.SQLImplicits

object jsoncsv {

def main(args: Array[String]) = {

val spark = SparkSession

.builder()

.appName("jsoncsv")

.config("spark.sql.warehouse.dir","/warehouse/tablespace/managed/hive")

.config("hive.metastore.uris", "thrift://hdp2.infocepts.com:9083")

.master("local")

.enableHiveSupport()

.getOrCreate()

var a = args(0)

var b = args(1)

val jsonfile = "/user/bctfeb20-bigdata-jaskeerat.s/"+a

val csvfile = "zomato\_etl/source/"+b

val rdd = spark.read.format("json").load(jsonfile)

val rdd1 = rdd.select(explode(col("restaurants.restaurant")))

val rdd2 = rdd1.select(col("col.R.res\_id") as "Restaurant Id",col("col.name") as "Restaurant Name",col("col.location.country\_id") as "Country Code",col("col.location.city") as "City",col("col.location.address") as "Address",col("col.location.locality") as "Locality",col("col.location.locality\_verbose") as "Locality Verbose",col("col.location.longitude") as "Longitude",col("col.location.latitude") as "Latitude",col("col.cuisines") as "Cuisines",col("col.average\_cost\_for\_two") as "Average Cost for two",col("col.currency") as "Currency",col("col.has\_table\_booking") as "Has Table booking",col("col.has\_online\_delivery") as "Has Online delivery",col("col.is\_delivering\_now") as "Is delivering now",col("col.switch\_to\_order\_menu") as "Switch to order menu",col("col.price\_range") as "Price range", col("col.user\_rating.aggregate\_rating") as "Aggregate rating",col("col.user\_rating.rating\_text") as "Rating text",col("col.user\_rating.votes") as "Votes")

rdd2.write.option("header","true").csv(csvfile)

spark.stop()

}

}

**SHELL SCRIPT to run the spark application**

To run the shell script

Make a empty file: **touch shell.sh**

Open the file: **vi shell.sh**

Write the code:

#!bin/bash

echo "Enter json file name"

read c

echo "Enter json file name"

read d

spark-submit --master yarn --deploy-mode "client" --name "testingApplication" --driver-memory 512m --driver-cores 1 --num-executors 3 --executor-cores 1 --executor-memory 512m --class jsoncsv /home/bctfeb20-bigdata-jaskeerat.s/test\_project\_jaskeerat/target/scala-2.11/test\_project\_jaskeerat\_2.11-1.0.jar $c $d

Make the file executable: **Chmod u+x shell.sh**

Execute the shell script: **source shell.sh**

**MODULE 2**

import java.text.SimpleDateFormat

import com.typesafe.config.{Config, ConfigFactory}

import java.io.{File, InputStream}

import java.net.URLDecoder

import java.security.MessageDigest

import java.util.Calendar

import org.apache.commons.net.util.SubnetUtils

import org.apache.hadoop.fs.{FileSystem,Path}

import org.apache.spark.graphx.{EdgeDirection, \_}

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.{StructField, StructType}

import java.io.PrintWriter

import org.apache.spark.sql.functions.explode

import org.apache.spark.sql.SparkSession

import org.apache.spark.sql.functions.col

import org.apache.spark.sql.SQLImplicits

object jsoncsv {

def main(args: Array[String]) = {

val spark = SparkSession

.builder()

.appName("jsoncsv")

.config("spark.sql.warehouse.dir","/warehouse/tablespace/managed/hive")

.config("hive.metastore.uris", "thrift://hdp2.infocepts.com:9083")

.master("local")

.enableHiveSupport()

.getOrCreate()

var a = args(0)

var b = args(1)

var c = args(2)

spark.sql(s"load data inpath '$a' into Table $b partition(filedate = $c)")

spark.stop()

}

}

**SHELL SCRIPT to run the spark application**

To run the shell script

Make a empty file: **touch load.sh**

Open the file: **vi load.sh**

**Write the code:**

#!bin/bash

echo "Enter the full path"

read a

echo "Enter the table name with database"

read b

if [ $b == "bct\_bi\_20.zomato\_jassi\_example" ]

then

echo "Enter the partition"

read c

spark-submit --master yarn --deploy-mode "client" --name "testingApplication" --driver-memory 512m --driver-cores 1 --num-executors 3 --executor-cores 1 --executor-memory 512m --class jsoncsv /home/bctfeb20-bigdata-jaskeerat.s/test\_project\_jaskeerat/target/scala-2.11/test\_project\_jaskeerat\_2.11-1.0.jar $a $b $c

else

echo "No Access"

fi

Make the file executable: **Chmod u+x load.sh**

Execute the shell script: **source load.sh**

**MODULE 3**

**CREATE A hive.hql file**

touch hive.hql

vi hive.hql

INSERT OVERWRITE table bct\_bi\_20.zomato\_summary\_jaskeerat\_managed

PARTITION(p\_filedate,p\_country\_name)

select

CAST(`Restaurant ID` as INT),

CAST(`Restaurant Name` as STRING),

CAST(zomato\_jassi\_source.`Country Code` as INT),

CAST(`City` as STRING),

CAST(`Address` as STRING),

CAST(`Locality` as STRING),

CAST(`Locality Verbose` as STRING),

CAST(`Longitude` as STRING),

CAST(`Latitude` as STRING),

CAST(`Cuisines` as STRING),

CAST(`Average Cost for two` as INT),

CAST(`Currency` as STRING),

CAST(`Has Table booking` as INT),

CAST(`Has Online delivery` as INT),

CAST(`Is delivering now` as INT),

CAST(`Switch to order menu` as INT),

CAST(`Price range` as INT),

CAST(`Aggregate rating` as DOUBLE),

CAST(`Rating text` as STRING),

CAST(`Votes` as INT),

null,

null,

FROM\_UNIXTIME( UNIX\_TIMESTAMP()),

CURRENT\_USER(),

CAST(zomato\_jassi\_source.`filedate` as INT),

CAST(dim\_country\_jassi.`Country` as STRING)

from bct\_bi\_20.zomato\_jassi\_source,bct\_bi\_20.dim\_country\_jassi

where zomato\_jassi\_source.`Country Code`=dim\_country\_jassi.`Country code`;

UPDATE bct\_bi\_20.zomato\_summary\_jaskeerat\_managed SET `m\_rating\_colour` = (

CASE WHEN (`rating text` = "Poor" and (`aggregate rating` >=1.9 and `aggregate rating` <=2.4)) THEN "Red"

WHEN (`rating text` = "Average" and (`aggregate rating` >=2.5 and `aggregate rating` <=3.4)) THEN "Amber"

WHEN (`rating text` = "Good" and (`aggregate rating` >=3.5 and `aggregate rating` <=3.9)) THEN "Light Green"

WHEN (`rating text` = "Very Good" and (`aggregate rating` >=4.0 and `aggregate rating` <=4.4)) THEN "Green"

WHEN (`rating text` = "Excellent" and (`aggregate rating` >=4.5 and `aggregate rating` <=5.0)) THEN "Gold"

ELSE 'NA' END);

UPDATE bct\_bi\_20.zomato\_summary\_jaskeerat\_managed SET `m\_cuisines` = (

CASE

WHEN (`cuisines` like '%Assamese%') THEN "Indian"

WHEN (`cuisines` like '%Andhra%') THEN "Indian"

WHEN (`cuisines` like '%Awadhi%') THEN "Indian"

WHEN (`cuisines` like '%Bengali%') THEN "Indian"

WHEN (`cuisines` like '%Bihari%') THEN "Indian"

WHEN (`cuisines` like '%Biryani%') THEN "Indian"

WHEN (`cuisines` like '%Chettinad%') THEN "Indian"

WHEN (`cuisines` like '%Curry%') THEN "Indian"

WHEN (`cuisines` like '%Gujrati%') THEN "Indian"

WHEN (`cuisines` like '%Goan%') THEN "Indian"

WHEN (`cuisines` like '%Hyderabadi%') THEN "Indian"

WHEN (`cuisines` like '%Indian%') THEN "Indian"

WHEN (`cuisines` like '%Kashmiri%') THEN "Indian"

WHEN (`cuisines` like '%Kerala%') THEN "Indian"

WHEN (`cuisines` like '%Lucknowi%') THEN "Indian"

WHEN (`cuisines` like '%Maharashtrian%') THEN "Indian"

WHEN (`cuisines` like '%Modern Indian%') THEN "Indian"

WHEN (`cuisines` like '%Mangalorean%') THEN "Indian"

WHEN (`cuisines` like '%Mughlai%') THEN "Indian"

WHEN (`cuisines` like '%Malwani%') THEN "Indian"

WHEN (`cuisines` like '%Mithai%') THEN "Indian"

WHEN (`cuisines` like '%Naga%') THEN "Indian"

WHEN (`cuisines` like '%North Indian%') THEN "Indian"

WHEN (`cuisines` like '%Oriya%') THEN "Indian"

WHEN (`cuisines` like '%Rajasthani%') THEN "Indian"

WHEN (`cuisines` like '%South Indian%') THEN "Indian"

WHEN (`cuisines` like '%Parsi%') THEN "Indian"

WHEN (`cuisines` = "North Indian") THEN "Indian"

WHEN (`cuisines` = "South Indian") THEN "Indian"

WHEN (`cuisines` = "Modern Indian") THEN "Indian"

ELSE 'World Cuisines' END);

UPDATE bct\_bi\_20.zomato\_summary\_jaskeerat\_managed SET `cuisines` = (

CASE

WHEN `cuisines` is null THEN "NA"

ELSE `cuisines` END);

RUNNING THE SHELL SCRIPT FOR hivefile.hql FILE:

#!bin/bash

echo 'Running the hivefile.hql file'

hive -f hivefile.hql

echo 'all the transformations are successful'

TO RUN THE hivefile.hql FILE:

Touch hivefile.sh

Vi hivefile.sh

Chmod u+x hivefile.sh

Source hivefile.sh

**Q6**

MOVE THE FILES INTO THE ARCHIVE FOLDER:

hdfs dfs -mv file1.json zomato\_etl/archive

hdfs dfs -mv file2.json zomato\_etl/archive

hdfs dfs -mv file3.json zomato\_etl/archive

hdfs dfs -ls zomato\_etl/archive

**Q7**

hdfs dfs -put file4.json zomato\_etl/source/json

hdfs dfs -put file5.json zomato\_etl/source/json

**To run the shell script**

Make a empty file: **touch shell.sh**

Open the file: **vi shell.sh**

Write the code:

#!bin/bash

echo "Enter json file name"

read c

echo "Enter json file name"

read d

spark-submit --master yarn --deploy-mode "client" --name "testingApplication" --driver-memory 512m --driver-cores 1 --num-executors 3 --executor-cores 1 --executor-memory 512m --class jsoncsv /home/bctfeb20-bigdata-jaskeerat.s/test\_project\_jaskeerat/target/scala-2.11/test\_project\_jaskeerat\_2.11-1.0.jar $c $d

Make the file executable: **Chmod u+x shell.sh**

Execute the shell script: **source shell.sh**

**To run the shell script**

Make a empty file: **touch load.sh**

Open the file: **vi load.sh**

Write the code:

#!bin/bash

echo "Enter the full path"

read a

echo "Enter the table name with database"

read b

echo "Enter the partition"

read c

spark-submit --master yarn --deploy-mode "client" --name "testingApplication" --driver-memory 512m --driver-cores 1 --num-executors 3 --executor-cores 1 --executor-memory 512m --class jsoncsv /home/bctfeb20-bigdata-jaskeerat.s/test\_project\_jaskeerat/target/scala-2.11/test\_project\_jaskeerat\_2.11-1.0.jar $a $b $c

Make the file executable: **Chmod u+x load.sh**

Execute the shell script: **source load.sh**

**Q8**

**MAKING CRONTAB**

Crontab -e

0 1 \* \* \* /bin/bash /home/bctfeb20-bigdata-jaskeerat.s/shell.sh

**Q9**

#table zomato\_jassi\_source

#total number of rows

select count(\*) from zomato\_jassi\_example;

29765 rows

#total number of columns

SELECT COUNT(\*) FROM INFORMATION\_SCHEMA.COLUMNS WHERE table\_name = 'zomato\_jassi\_example';

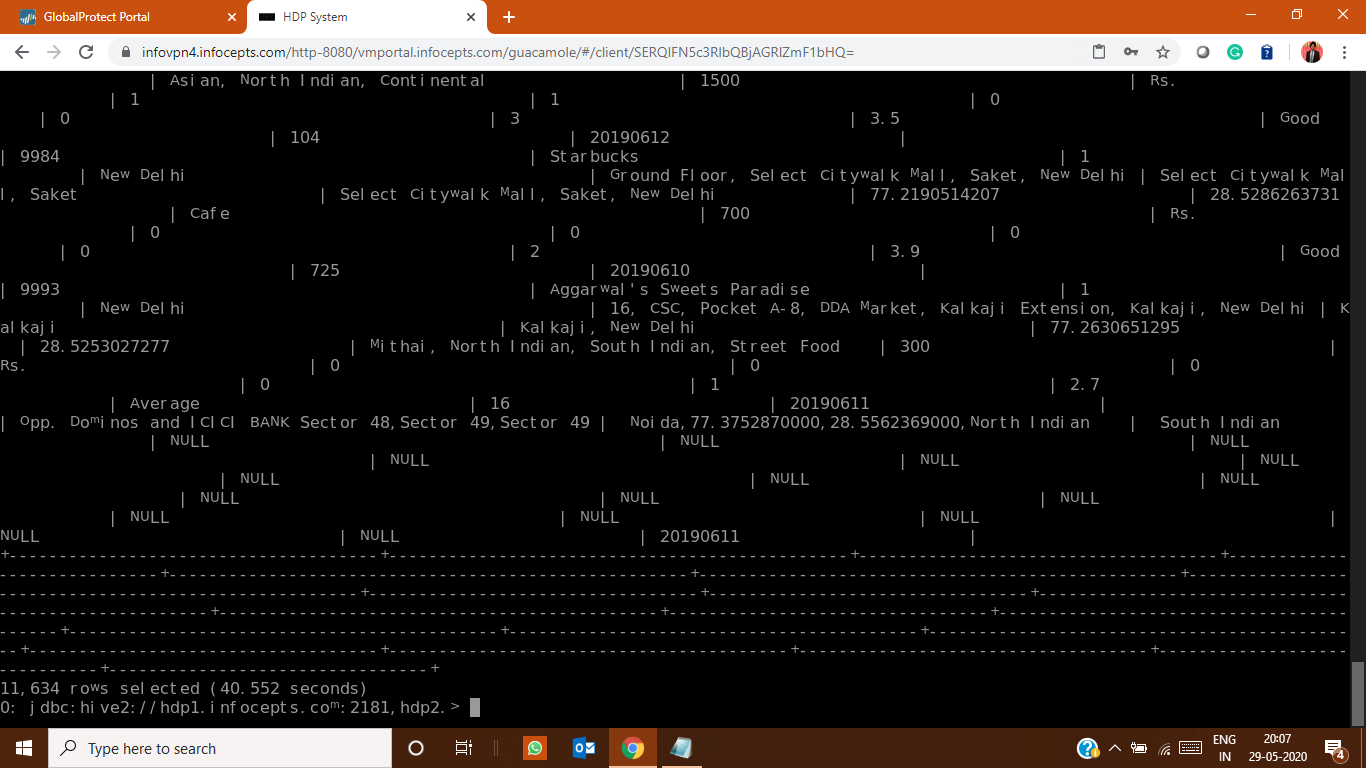
describe zomato\_jassi\_example;

21

#getting distinct rows

select distinct \* from zomato\_jassi\_source

11634 rows



#number of duplicates

29765-11634=18131

#duplicate check

select `restaurant id`, `restaurant name`, `country code`, city, address, locality,

`locality verbose`, longitude, latitude, cuisines, `average cost for two`, currency, `has table booking`,

`has online delivery`, `is delivering now`, `switch to order menu`, `price range`,

`aggregate rating`, `rating text`, votes, count(\*)

from zomato\_jassi\_example

group by `restaurant id`, `restaurant name`, `country code`, city, address, locality,

`locality verbose`, longitude, latitude, cuisines, `average cost for two`, currency, `has table booking`,

`has online delivery`, `is delivering now`, `switch to order menu`, `price range`,

`aggregate rating`, `rating text`, votes

having count(\*) > 1;

2455

=========================================================================================

#table dim\_country\_jassi

#total number of rows

select count(\*) from dim\_country\_jassi;

17

#total number of columns

describe dim\_country\_jassi

2

#getting distinct rows

select distinct \* from dim\_country\_jassi;

17

#numbers of null

select count(\*) from dim\_country\_jassi where `country code` is null OR `country` is null;

2

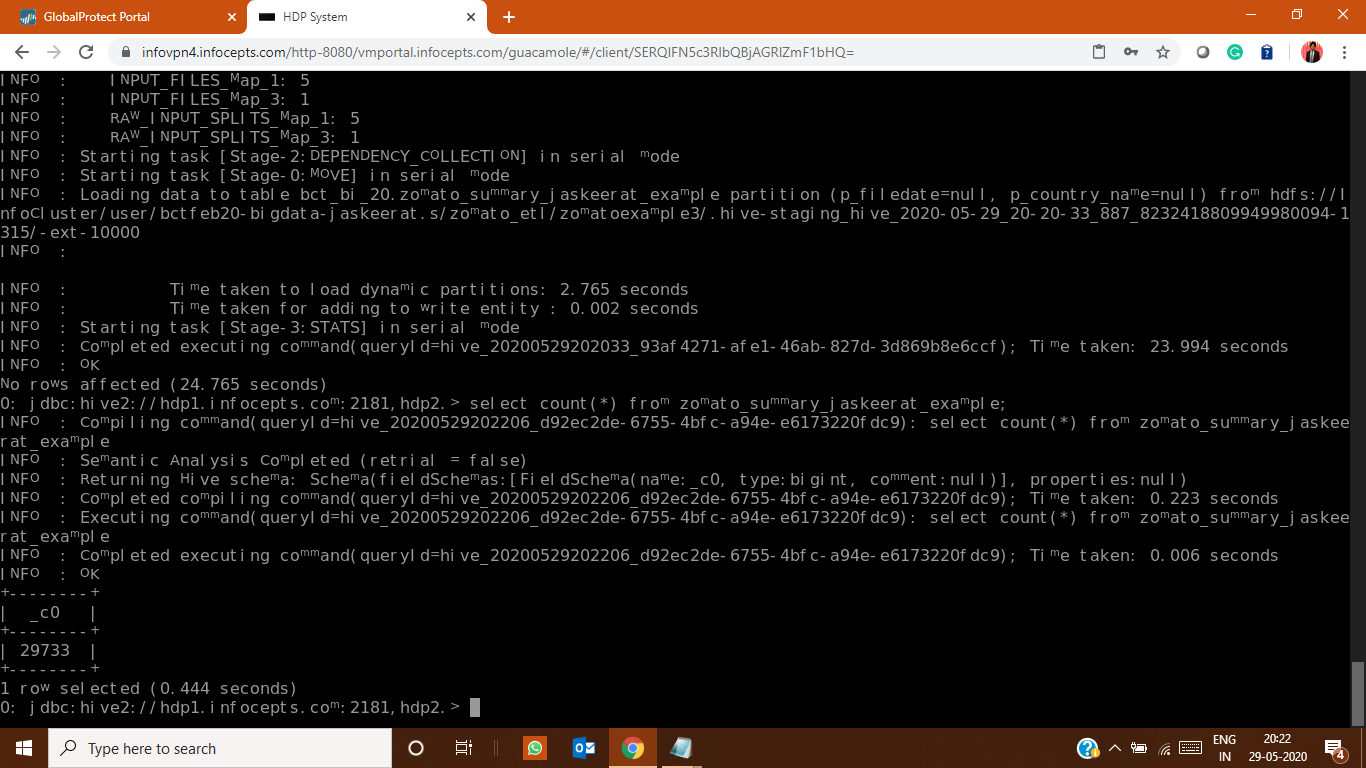
========================================================================================

#table zomato\_summary\_jaskeerat\_managed and zomato\_summary\_jaskeerat\_example

#total number of rows

select count(\*) from zomato\_summary\_jaskeerat\_example;

29733

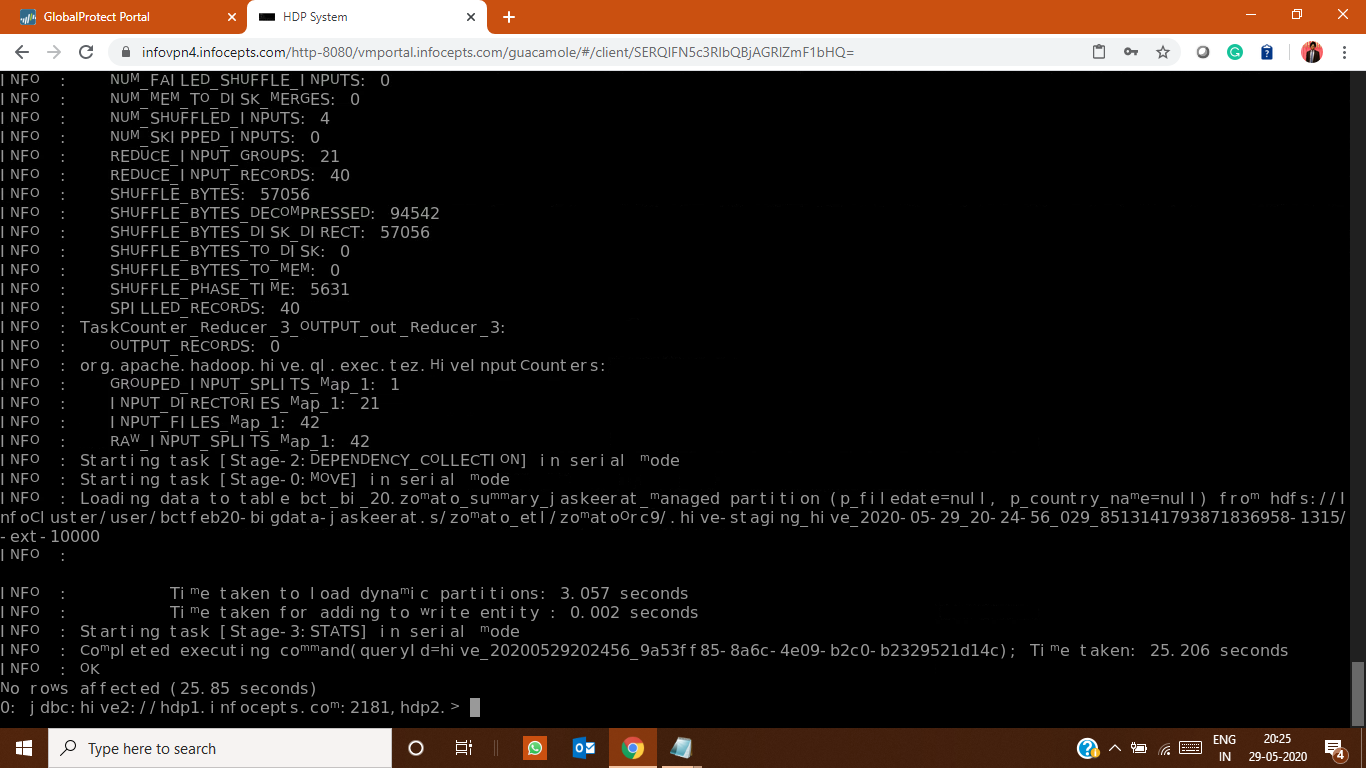


#total number of columns

describe zomato\_summary\_jaskeerat\_example;

26

after running- insert overwrite table zomato\_summary\_jaskeerat\_managed select distinct \* from zomato\_summary\_jaskeerat\_managed;



#getting distinct rows

select distinct \* from zomato\_summary\_jaskeerat\_example;

11602



#getting nulls

select votes from zomato\_summary\_jaskeerat\_managed where votes is NULL;

6

UPDATE zomato\_summary\_jaskeerat\_managed SET `votes` = "NA" WHERE `votes` IS NULL;

select `locality` from zomato\_summary\_jaskeerat\_managed where `locality` is NULL;

6

UPDATE zomato\_summary\_jaskeerat\_managed SET `locality` = "NA" WHERE `locality` IS NULL;

select `locality verbose` from zomato\_summary\_jaskeerat\_managed where `locality verbose` is NULL;

6

UPDATE zomato\_summary\_jaskeerat\_managed SET `locality verbose` = "NA" WHERE `locality verbose` IS NULL;

select `address` from zomato\_summary\_jaskeerat\_managed where `address` is NULL;

6

UPDATE zomato\_summary\_jaskeerat\_managed SET `address` = "NA" WHERE `address` IS NULL;

select `longitude` from zomato\_summary\_jaskeerat\_managed where `longitude` is NULL;

6

UPDATE zomato\_summary\_jaskeerat\_managed SET `longitude` = "NA" WHERE `longitude` IS NULL;

select `latitude` from zomato\_summary\_jaskeerat\_managed where `latitude` is NULL;

6

UPDATE zomato\_summary\_jaskeerat\_managed SET `latitude` = "NA" WHERE `latitude` IS NULL;

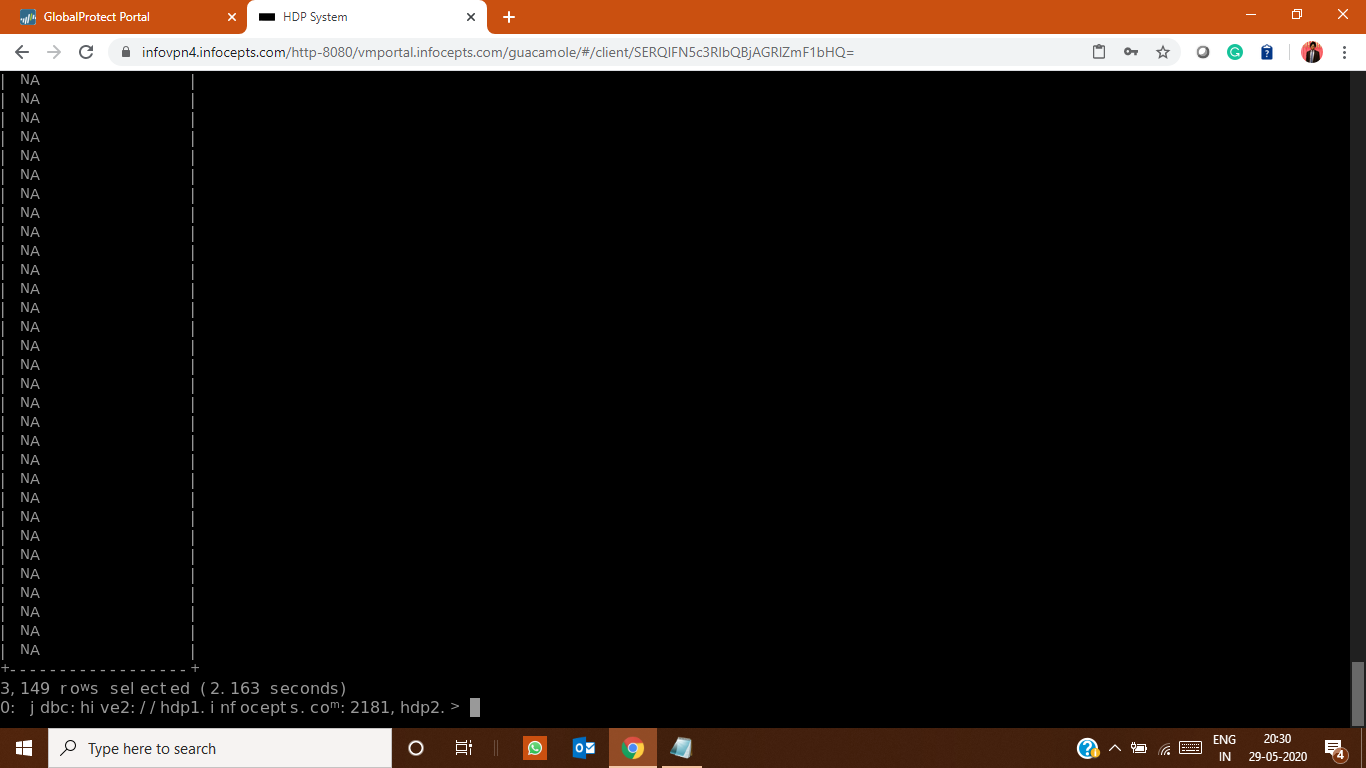
select `rating text` from zomato\_summary\_jaskeerat\_managed where `rating text` is NULL;

6

UPDATE zomato\_summary\_jaskeerat\_managed SET `rating text` = "NA" WHERE `rating text` IS NULL;

#anomalies in m\_rating\_colour coloumn

select m\_rating\_colour from zomato\_summary\_jaskeerat\_example where m\_rating\_colour="NA";

3149

**Data column validation of data Input from source table to summary table:**

|  |  |  |  |
| --- | --- | --- | --- |
| zomato\_jassi\_source | zomato\_summary\_jaskeerat\_example | True/False | True/False |
| `Restaurant ID` | `Restaurant ID` | TRUE | TRUE |
| `Restaurant Name` | `Restaurant Name` | TRUE | TRUE |
| `Country Code` | `Country Code` | TRUE | TRUE |
| `City` | `City` | TRUE | TRUE |
| `Address` | `Address` | TRUE | TRUE |
| `Locality` | `Locality` | TRUE | TRUE |
| `Locality Verbose` | `Locality Verbose` | TRUE | TRUE |
| `Longitude` | `Longitude` | TRUE | TRUE |
| `Latitude` | `Latitude` | TRUE | TRUE |
| `Cuisines` | `Cuisines` | TRUE | TRUE |
| `Average Cost for two` | `Average Cost for two` | TRUE | TRUE |
| `Currency` | `Currency` | TRUE | TRUE |
| `Has Table booking` | `Has Table booking` | TRUE | TRUE |
| `Has Online delivery` | `Has Online delivery` | TRUE | TRUE |
| `Is delivering now` | `Is delivering now` | TRUE | TRUE |
| `Switch to order menu` | `Switch to order menu` | TRUE | TRUE |
| `Price range` | `Price range` | TRUE | TRUE |
| `Aggregate rating` | `Aggregate rating` | TRUE | TRUE |
| `Rating text` | `Rating text` | TRUE | TRUE |
| `Votes` | `Votes | TRUE | TRUE |

**Data validation for comparing the country code on both table:**

|  |  |  |  |
| --- | --- | --- | --- |
| dim\_country\_jassi | zomato\_summary\_jaskeerat\_example | True/False | True/Faslse |
| 1 : India | 1 : India | True | True |
| 14 : Australia | 14 : Australia | True | True |
| 30 : Brazil | 30 : Brazil | True | True |
| 37 : Canada | 37 : Canada | True | True |
| 94 : Indonesia | 94 : Indonesia | True | True |
| 148 : New Zealand | 148 : New Zealand | True | True |
| 162 : Phillipines | 162 : Phillipines | True | True |
| 166 : Qatar | 166 : Qatar | True | True |
| 184 : Singapore | 184 : Singapore | True | True |
| 189 : South Africa | 189 : South Africa | True | True |
| 191 : Sri Lanka | 191 : Sri Lanka | True | True |
| 208 : Turkey | 208 : Turkey | True | True |
| 214 : UAE | 214 : UAE | True | True |
| 215 : United Kingdom | 215 : United Kingdom | True | True |
| 216 : United States | 216 : United States | True | True |

**Q10**

CREATE TABLE bct\_bi\_20.zomato\_summary\_log\_jassi(

`Job id` STRING,

`Job Step` STRING,

`Spark submit command` STRING,

`Job Start time` STRING,

`Job End time` STRING,

`Job status` STRING

)

ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'

WITH SERDEPROPERTIES (

"separatorChar" = ","

)STORED AS TEXTFILE;

load data inpath '/user/bctfeb20-bigdata-jaskeerat.s/logs.csv' into table zomato\_summary\_log\_jassi;

**Application ID : application\_1589956236740\_4967**

Start-Time : 1590758073294

Finish-Time : 1590758484235

Execution-Time : 410941

NO of executors-2

NO of cores-1

Driver memory-1gb

Tracking-URL : http://hdp1.infocepts.com:8088/cluster/app/application\_1589956236740\_4967

**Application ID : application\_1589956236740\_4968**

Start-Time : 1590758844653

Finish-Time : 1590759048953

Execution -Time: 204300

NO of executors-2

NO of cores-1

Driver memory-1gb

Tracking-URL : http://hdp1.infocepts.com:8088/cluster/app/application\_1589956236740\_4968

**Application ID : application\_1589956236740\_4969**

Start-Time : 1590759175059

Finish-Time : ,1590759373550

Execution-Time:198491

NO of executors-2

NO of cores-1

Driver memory-1gb

Tracking-URL : http://hdp1.infocepts.com:8088/cluster/app/application\_1589956236740\_4969

**Application ID : application\_1589956236740\_4970**

Start-Time : 1590759488755

Finish-Time : 1590759586458

Execution-Time:97703

NO of executors-2

NO of cores-1

Driver memory-1gb

Tracking-URL :

<http://hdp1.infocepts.com:8088/cluster/app/application_1589956236740_4970>

**Application ID : application\_1589956236740\_4971**

Start-Time : 1590759679412

Finish-Time : 1590760678792

Execution-Time:999380

NO of executors-2

NO of cores-1

Driver memory-1gb

Tracking-URL : http://hdp1.infocepts.com:8088/cluster/app/application\_1589956236740\_4971

**Application ID : application\_1589956236740\_4972**

Start-Time : 1590761122450

Finish-Time : 1590761202449

Execution-Time:79999

NO of executors-2

NO of cores-1

Driver memory-1gb

Tracking-URL : http://hdp1.infocepts.com:8088/cluster/app/application\_1589956236740\_4972

**Application ID : application\_1589956236740\_4973**

Start-Time : 1590761323151

Finish-Time : 1590761452569

Execution-Time: 129418

NO of executors-2

NO of cores-1

Driver memory-1gb

Tracking-URL : http://hdp1.infocepts.com:8088/cluster/app/application\_1589956236740\_4973

**Application ID : application\_1589956236740\_4975**

Start-Time : 1590761624696

Finish-Time : 1590761764876

Execution-Time: 140180

NO of executors-2

NO of cores-1

Driver memory-1gb

Tracking-URL : http://hdp1.infocepts.com:8088/cluster/app/application\_1589956236740\_4975

**Application ID : application\_1589956236740\_4976**

Start-Time : 1590761835204

Finish-Time : 1590761906936

Execution-Time: 71732

NO of executors-2

NO of cores-1

Driver memory-1gb

Tracking-URL : http://hdp1.infocepts.com:8088/cluster/app/application\_1589956236740\_4976

**Application ID : application\_1589956236740\_4977**

Start-Time : 1590762018054

Finish-Time : 1590766258959

Execution-Time: 4240905

NO of executors-2

NO of cores-1

Driver memory-1gb

Tracking-URL : http://hdp1.infocepts.com:8088/cluster/app/application\_1589956236740\_4977

**Q11**

A file for unit testing is prepared separately.

**Q12**

All the guidelines which are mentioned in the project are followed.

**Q13**

hdfs dfs -mkdir zomato\_etl

hdfs dfs -mkdir zomato\_etl/source

hdfs dfs -mkdir zomato\_etl/source/csv

hdfs dfs -mkdir zomato\_etl/source/json

hdfs dfs -mkdir zomato\_etl/script

hdfs dfs -mkdir zomato\_etl/archive

hdfs dfs -put file1.json zomato\_etl/source/json

hdfs dfs -put file2.json zomato\_etl/source/json

hdfs dfs -put file3.json zomato\_etl/source/json

hdfs dfs -put file4.json zomato\_etl/source/json

hdfs dfs -put file5.json zomato\_etl/source/json

hdfs dfs -put zomato\_20190609.csv zomato\_etl/source/csv

hdfs dfs -put zomato\_20190610.csv zomato\_etl/source/csv

hdfs dfs -put zomato\_20190611.csv zomato\_etl/source/csv

hdfs dfs -put zomato\_20190612.csv zomato\_etl/source/csv

hdfs dfs -put zomato\_20190613.csv zomato\_etl/source/csv

hdfs dfs -put load.sh zomato\_etl/script

hdfs dfs -put shell.sh zomato\_etl/script

**Q14**

hdfs dfs -mkdir zomato\_etl\_jaskeerat

hdfs dfs -mkdir zomato\_etl\_jaskeerat/log

hdfs dfs -mkdir zomato\_etl\_jaskeerat/zomato\_ext

hdfs dfs -mkdir zomato\_etl\_jaskeerat/zomato\_ext/zomato

hdfs dfs -mkdir zomato\_etl\_jaskeerat/zomato\_ext/dim\_country

All the guidelines which are mentioned in the project report are followed.

**MAKING SBT APPLICATION**

Make any directory

Make a build.sbt file

To add all the folder structure including project,target run “sbt clean”

ADD all the dependancies

name :="test\_project\_jaskeerat"

version := "1.0"

scalaVersion := "2.11.8"

libraryDependencies += "org.apache.spark" %% "spark-hive" % "2.3.4" % "provided"

libraryDependencies += "org.apache.spark" %% "spark-core" % "2.3.4"

libraryDependencies += "org.apache.spark" %% "spark-sql" % "2.3.4"

libraryDependencies += "com.typesafe" % "config" % "1.4.0"

libraryDependencies += "org.apache.spark" %% "spark-graphx" % "2.3.4"

In the same directory make a .scala file

To make a JAR after making .scala file then run “sbt package”

Run the JAR using spark-submit with all the parameters