Task 1

```scala
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.types._
import org.apache.spark.sql.Row
import org.apache.spark.sql.functions._
import org.apache.spark._
import scala.io.Source
import com.mysql.jdbc.Driver

object Task1 {
  def main(args:Array[String]):Unit={

      val conf = new SparkConf().setAppName("zipcode").setMaster("local[*]")
          val sc   = new SparkContext(conf)
          sc.setLogLevel("Error")

          val spark = SparkSession.builder().getOrCreate()
          import spark.implicits._

          //Get Buffer response
          val http_buffer = Source.fromURL("https://randomuser.me/api/0.8/?results=10")

          //Create String from buffer
          val http_string = http_buffer.mkString

          //Create rdd[String] from String
          val rdd_str = sc.parallelize(List(http_string))

          //Convert rdd to data frame
          val df_http = spark.read.json(rdd_str)
          //df_http.show()
```

```scala
          //Convert rdd to data frame
          val df_http = spark.read.json(rdd_str)
          //df_http.show()
          df_http.printSchema()

          val df_http_all = df_http.withColumn("results", explode(col("results")))
          .select("results.user.gender","results.user.name.title", "results.user.name.first",
              "results.user.name.last","results.user.location.street",
              "results.user.location.city","results.user.location.state",
              "results.user.location.zip","results.user.email",
              "results.user.username", "results.user.password",
              "results.user.salt","results.user.md5",
              "results.user.sha1","results.user.sha256","results.user.registered","results.user.dob
              "results.user.phone","results.user.cell",
              "results.user.picture.large",
              "results.user.picture.medium","results.user.picture.thumbnail",
              "nationality","seed","version")

          //write to hive
          df_http_all.write.mode("overwrite").format("orc").saveAsTable("zeyodb.userdata")

          println("data written to hive successfully")

          val df_hive_read = spark.sql("select * from zeyodb.userdata")
          df_hive_read.show()

          println("data read from hive successfully")

          df_http_all.write.format("jdbc").option("url","jdbc:mysql://localhost/zeyodb")
          .mode("overwrite")
          .option("driver", "com.mysql.jdbc.Driver")
          .option("dbtable","userdata")
          .option("user","dbuser")
```

```
        //write to hive
        df_http_all.write.mode("overwrite").format("orc").saveAsTable("zeyodb.userdata")

        println("data written to hive successfully")

        val df_hive_read = spark.sql("select * from zeyodb.userdata")
        df_hive_read.show()

        println("data read from hive successfully")

        df_http_all.write.format("jdbc").option("url","jdbc:mysql://localhost/zeyodb")
        .mode("overwrite")
        .option("driver", "com.mysql.jdbc.Driver")
        .option("dbtable","userdata")
        .option("user","dbuser")
        .option("password","cloudera").save()
        println("Data Written  to RDBMS")

        val df_read_rdbms = spark.read.format("jdbc").option("url","jdbc:mysql://localhost/zeyodb
        .option("driver", "com.mysql.jdbc.Driver")
        .option("dbtable","userdata")
        .option("user","dbuser")
        .option("password","cloudera").load()

        df_read_rdbms.write.mode("overwrite").format("parquet").saveAsTable("zeyodb.parquet")
        println("Data written to hive")
    }

}
```

Task 2

```
//Create rdd[String] from String
val rdd_str = sc.parallelize(List(http_string))

//Convert rdd to data frame
val df_http = spark.read.json(rdd_str)
//df_http.show()
df_http.printSchema()

val df_http_all = df_http.withColumn("results", explode(col("results")))
.select("results.user.location.zip")
//df_http_all.show()
//Overwrite for the first time and append other times
var write_mode = ""
if(i == 1)
    write_mode = "overwrite"
else
    write_mode = "append"

df_http_all.write.format("jdbc").option("url","jdbc:mysql://localhost/zeyodb")
.mode(write_mode)
.option("driver", "com.mysql.jdbc.Driver")
.option("dbtable","zipcode")
.option("user","dbuser")
.option("password","cloudera").save()
println("Written - " + i)
```

Problems Tasks Console

```
<terminated> Task2$ [Scala Application] C:\Program Files\Java\jdk1.8.0_172\bin\javaw.exe (12-Jun-2020, 8:49:38 AM)
 |    |    |    |-- sha1: string (nullable = true)
 |    |    |    |-- sha256: string (nullable = true)
 |    |    |    |-- username: string (nullable = true)
 |-- seed: string (nullable = true)
 |-- version: string (nullable = true)

Written - 10
```

```
Administrator: Command Prompt - mysql  -u dbuser -p
| 56408 |
| 90097 |
| 87071 |
| 93902 |
| 34441 |
| 79059 |
| 52816 |
| 75107 |
| 48591 |
| 24021 |
| 17448 |
| 51854 |
| 86805 |
| 43392 |
| 62958 |
| 19780 |
| 32671 |
| 59670 |
| 65765 |
| 27220 |
| 79272 |
| 34826 |
| 63728 |
| 58609 |
| 46654 |
| 66121 |
+-------+
100 rows in set (0.00 sec)

mysql>
```

Task 3

```
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.types._
import org.apache.spark.sql.Row
import org.apache.spark.sql.functions._
import org.apache.spark._
import scala.io.Source
import com.mysql.jdbc.Driver

object Task3 {
   def main(args:Array[String]):Unit={

      val conf = new SparkConf().setAppName("zipcode").setMaster("local[*]")
          val sc   = new SparkContext(conf)
          sc.setLogLevel("Error")

          val spark = SparkSession.builder().enableHiveSupport().getOrCreate()
          import spark.implicits._
          val repeat_count = 10
          for(i <- 1 to repeat_count)
          {
                //Get Buffer response
                val http_buffer = Source.fromURL("https://randomuser.me/api/0.8/?results=10")

                //Create String from buffer
                val http_string = http_buffer.mkString

                //Create rdd[String] from String
                val rdd_str = sc.parallelize(List(http_string))

                //Convert rdd to data frame
                val df_http = spark.read.json(rdd_str)
                //df_http.show()
```

```
                val rdd_str = sc.parallelize(List(http_string))

                //Convert rdd to data frame
                val df_http = spark.read.json(rdd_str)
                //df_http.show()
                df_http.printSchema()

                val df_http_all = df_http.withColumn("results", explode(col("results")))
                .select("results.user.location.zip")
                //df_http_all.show()
                //Overwrite for the first time and append other times
                var write_mode = ""
                if(i == 1)
                    write_mode = "overwrite"
                else
                    write_mode = "append"

                //Write to RDBMS
                df_http_all.write.format("jdbc").option("url","jdbc:mysql://localhost/zeyodb")
                .mode(write_mode)
                .option("driver", "com.mysql.jdbc.Driver")
                .option("dbtable","zipcode")
                .option("user","dbuser")
                .option("password","cloudera").save()
                println("Written - " + i)
                println("data written to rdbms")

                //Write to Hive
                df_http_all.write.format("hive").mode(write_mode).saveAsTable("zipcode")
                println("data written to hive")
            }
        }
}
```

Command to submit job:

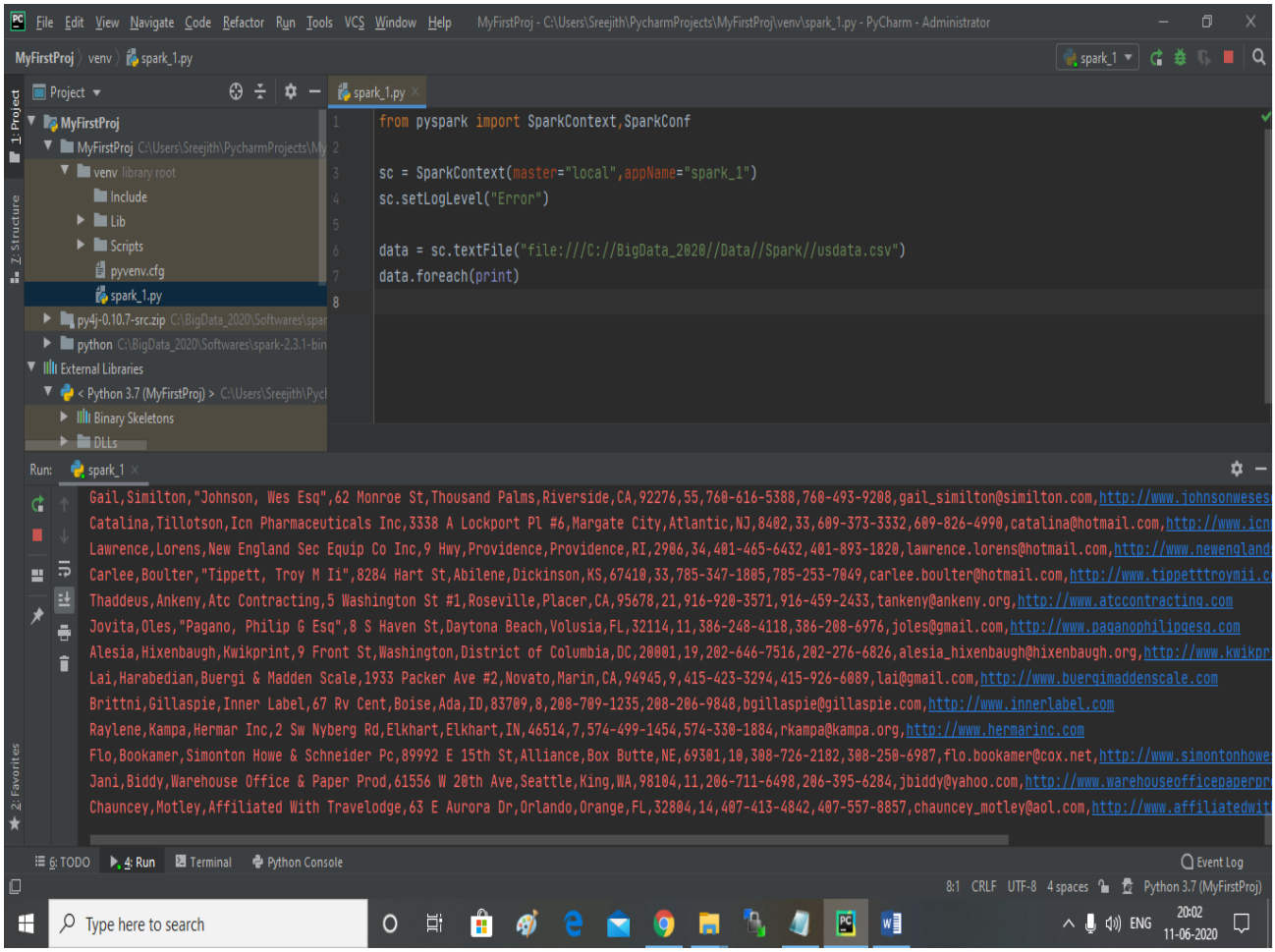
spark-submit --class Task.Task --master local[*] --jars "/home/cloudera/ext_jars/*" --conf spark.sql.catalogImplementation=hive /home/cloudera/program_jars/Task-0.0.1-SNAPSHOT.jar

Task 4

Scheduling the job using cron tab:

/20 * * * * spark-submit --class Task.Task --master local[*] --jars "/home/cloudera/ext_jars/*" --conf spark.sql.catalogImplementation=hive /home/cloudera/program_jars/Task-0.0.1-SNAPSHOT.jar

Task 5


File Edit View Navigate Code Refactor Run Tools VCS Window Help
MyFirstProj - C:\Users\Sreejith\PycharmProjects\MyFirstProj\venv\spark_1.py - PyCharm - Administrator
MyFirstProj venv spark_1.py
spark_1
Project
spark_1.py
MyFirstProj
venv library root
Include
Lib
Scripts
pyvenv.cfg
spark_1.py
py4j-0.10.7-src.zip
python
External Libraries
Binary Skeletons
DLLs
from pyspark import SparkContext,SparkConf
sc = SparkContext(master="local",appName="spark_1")
sc.setLogLevel("Error")
data = sc.textFile("file:///C://BigData_2020//Data//Spark//usdata.csv")
data.foreach(print)
Run: spark_1
Gail,Similton,"Johnson, Wes Esq",62 Monroe St,Thousand Palms,Riverside,CA,92276,55,760-616-5388,760-493-9208,gail_similton@similton.com,http://www.johnsonweses
Catalina,Tillotson,Icn Pharmaceuticals Inc,3338 A Lockport Pl #6,Margate City,Atlantic,NJ,8402,33,609-373-3332,609-826-4990,catalina@hotmail.com,http://www.icn
Lawrence,Lorens,New England Sec Equip Co Inc,9 Hwy,Providence,Providence,RI,2906,34,401-465-6432,401-893-1820,lawrence.lorens@hotmail.com,http://www.newengland
Carlee,Boulter,"Tippett, Troy M Ii",8284 Hart St,Abilene,Dickinson,KS,67410,33,785-347-1805,785-253-7049,carlee.boulter@hotmail.com,http://www.tippetttroymii.c
Thaddeus,Ankeny,Atc Contracting,5 Washington St #1,Roseville,Placer,CA,95678,21,916-920-3571,916-459-2433,tankeny@ankeny.org,http://www.atccontracting.com
Jovita,Oles,"Pagano, Philip G Esq",8 S Haven St,Daytona Beach,Volusia,FL,32114,11,386-248-4118,386-208-6976,joles@gmail.com,http://www.paganophilipgesq.com
Alesia,Hixenbaugh,Kwikprint,9 Front St,Washington,District of Columbia,DC,20001,19,202-646-7516,202-276-6826,alesia_hixenbaugh@hixenbaugh.org,http://www.kwikpr
Lai,Harabedian,Buergi & Madden Scale,1933 Packer Ave #2,Novato,Marin,CA,94945,9,415-423-3294,415-926-6089,lai@gmail.com,http://www.buergimaddenscale.com
Brittni,Gillaspie,Inner Label,67 Rv Cent,Boise,Ada,ID,83709,8,208-709-1235,208-206-9848,bgillaspie@gillaspie.com,http://www.innerlabel.com
Raylene,Kampa,Hermar Inc,2 Sw Nyberg Rd,Elkhart,Elkhart,IN,46514,7,574-499-1454,574-330-1884,rkampa@kampa.org,http://www.hermarinc.com
Flo,Bookamer,Simonton Howe & Schneider Pc,89992 E 15th St,Alliance,Box Butte,NE,69301,10,308-726-2182,308-250-6987,flo.bookamer@cox.net,http://www.simontonhowe
Jani,Biddy,Warehouse Office & Paper Prod,61556 W 20th Ave,Seattle,King,WA,98104,11,206-711-6498,206-395-6284,jbiddy@yahoo.com,http://www.warehouseofficepaperpr
Chauncey,Motley,Affiliated With Travelodge,63 E Aurora Dr,Orlando,Orange,FL,32804,14,407-413-4842,407-557-8857,chauncey_motley@aol.com,http://www.affiliatedwit
6: TODO
4: Run
Terminal
Python Console
Event Log
8:1 CRLF UTF-8 4 spaces
Python 3.7 (MyFirstProj)
Type here to search
ENG
20:02
11-06-2020

Task 6

```scala
object Task6 {

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

      val conf = new SparkConf().setAppName("DSL").setMaster("local[*]")
        val sc   = new SparkContext(conf)
        sc.setLogLevel("Error")

        val spark = SparkSession.builder().getOrCreate()
        import spark.implicits._

        val batters_df = spark.read.option("multiline",true).format("json").load("file:///C://BigData_2020//Data//Spark/batters.
        //batters_df.show()
        batters_df.printSchema()
        val batters_select = batters_df.withColumn("batter", explode(col("batters.batter")))
                                       .withColumn("topping", explode(col("topping")))
                                       .select(col("id"),col("name"),col("ppu"),col("type"),col("batter.id").alias("batter_id"),
                                       col("batter.type").alias("batter_type"),col("topping.id").alias("topping_id"),
                                       col("topping.type").alias("topping_type"))

        batters_select.show()
        batters_select.printSchema()
  }

}
```

Problems Tasks Console

<terminated> Task6$ [Scala Application] C:\Program Files\Java\jdk1.8.0_172\bin\javaw.exe (12-Jun-2020, 8:04:31 AM)

```
+----+----+----+-----+---------+----------+----------+-------------------+
|  id|name| ppu| type|batter_id|batter_type|topping_id|       topping_type|
+----+----+----+-----+---------+----------+----------+-------------------+
|0001|Cake|0.55|donut|     1001|   Regular|      5001|               None|
|0001|Cake|0.55|donut|     1001|   Regular|      5002|             Glazed|
|0001|Cake|0.55|donut|     1001|   Regular|      5005|              Sugar|
|0001|Cake|0.55|donut|     1001|   Regular|      5007|     Powdered Sugar|
```