CHECKPOINT 1

1. Uploading data into HDFS, Hive(internal), Hive(external) and Spark
    a. For HDFS, we just move the files from local to the HDFS environment by :

        hdfs dfs –put Project

    b. For Hive(internal), we run the following commands:

        i. create table if not exists aadhaar_details(registrar string,private_agency string,state string,district string,sub_district string,pincode string,gender string,age int,aadhaar_generated int,rejected int,provide_email int,provide_mobile int)

        row format delimited fields terminated by ','

        stored as textfile

        location "/user/cloudera/Project";

        ii. insert overwrite local directory '/home/cloudera/Project/Checkpoints/Checkpoints1'

        row format delimited fields terminated by ','

        stored as textfile

        select * from aadhaar_details LIMIT 25;

    c. For Hive(external), we run the following commands:

        i. create external table if not exists aadhaar_details_external(registrar string,private_agency string,state string,district string,sub_district

```
string,pincode string,gender string,age int,aadhaar_generated int,rejected
int,provide_email int,provide_mobile int)
row format delimited fields terminated by ','
stored as textfile
location "/user/cloudera/Project";
select * from aadhaar_details_external LIMIT 25;
```

d. For Spark, we use the following commands:

   i. `val aadhar_dets = sc.textFile("Project/aadhar.csv")`

   ii. `val first_header = aadhar_dets.first()`

   iii. `val final_details = aadhar_dets.filter(w=>w!=first_header)`

   iv.
```
val aadhar_details =
final_details.map(w=>(w.split(",")(0),w.split(",")(1),w.split(",")(2),w.split(
",")(3),w.split(",")(4),w.split(",")(5),w.split(",")(6),w.split(",")(7).toInt,w.s
plit(",")(8).toInt,w.split(",")(9).toInt,w.split(",")(10).toInt,w.split(",")(11).t
oInt))
```

   v.
```
aadhar_details.toDF("registrar","private_agency","state","district","sub_di
strict","pincode","gender","age","aadhaar_generated","rejected","noemails
","nomobile")
```

CHECKPOINT 2

2. The schema of the tables are shown by:

aadhar_DF.schema

3. The count and registrars are shown by:

sqlContext.sql("Select distinct(registrar) from Aadhar_Details").show()

sqlContext.sql("Select count(distinct(registrar)) from Aadhar_Details").show()

4. The following commands give the solution:

sqlContext.sql("Select state,count(district) from Aadhar_Details group by state").show()

sqlContext.sql("Select district,count(sub_district) from Aadhar_Details group by district").show()

5. The following commands give the solution:

sqlContext.sql("Select state,count(gender=='M') as Male,count(gender=='F') as Female from Aadhar_Details group by state").show()

CHECKPOINT 3

8. The following sequence of codes solves the problem:

sqlContext.sql("Select state,sum(aadhaar_generated) from Aadhar_Details group by state order by sum(aadhaar_generated)").show()

9. sqlContext.sql("Select private_agency,sum(aadhaar_generated) from Aadhar_Details group by private_agency order by sum(aadhaar_generated) limit 3").show()

10. sqlContext.sql("Select count(*) as Given from Aadhar_Details where noemails==1 and nomobile==1").show()

11. sqlContext.sql("Select district,sum(aadhaar_generated+rejected) as Enrollments from Aadhar_Details group by district order by sum(aadhaar_generated+rejected) desc limit 3").show()

12. sqlContext.sql("Select state,sum(aadhaar_generated) as Enrollments from Aadhar_Details where aadhaar_generated==1 group by state").show()

CHECKPOINT 4

13. The following code sequences solve the question:

```
 val aadhar_dets = sc.textFile("Project/aadhar.csv")

val first_header = aadhar_dets.first()

val final_details = aadhar_dets.filter(w=>w!=first_header)

val aadhar_DF =
aadhar_details.toDF("registrar","private_agency","state","district","sub_district","pincode","gen
der","age","aadhaar_generated","rejected","noemails","nomobile")

aadhar_DF.printSchema
```

14. The following code solves the problem:

```
aadhar_DF.select(corr('age,'nomobile)).show()
```

15. The following code solves the problem:

```
sqlContext.sql("Select count(distinct(pincode)) as PinCodes from
Aadhar_Details").show()
```

16. The following code solves the problem:

```
sqlContext.sql("Select state,sum(rejected) as Countaadhaar from Aadhar_Details where
state=='Uttar Pradesh' or state=='Maharashtra' group by state").show()
```

CHECKPOINT 5

17. The following code solves the problem:

```
sqlContext.sql("Select
state,sum(aadhaar_generated)/(sum(aadhaar_generated)+sum(rejected))*100 as
Percentaadhaar from Aadhar_Details where gender=='M' group by state order by
sum(aadhaar_generated)/(sum(aadhaar_generated)+sum(rejected))*100 DESC LIMIT
3").show()
```

18. The following sequence of codes solves the problem :

```
sqlContext.sql("Select
district,sum(aadhaar_generated)/(sum(aadhaar_generated)+sum(rejected))*100 as
Percentaadhaar from Aadhar_Details where gender=='F'and (state=='Others' or
state=='Lakshadweep' or state=='Andaman and Nicobar Islands') group by district order by
sum(aadhaar_generated)/(sum(aadhaar_generated)+sum(rejected))*100 DESC LIMIT
3").show()
```

19. The following sequence of codes solves the problem:

sqlContext.sql("Select state,sum(aadhaar_generated)/(sum(aadhaar_generated)+sum(rejected))*100 as Percentaadhaar from Aadhar_Details where gender=='F' group by state order by sum(aadhaar_generated)/(sum(aadhaar_generated)+sum(rejected))*100 DESC LIMIT 3").show()

20. The following set of codes solves the problem:

sqlContext.sql("Select district,sum(aadhaar_generated)/(sum(aadhaar_generated)+sum(rejected))*100 as Percentaadhaar from Aadhar_Details where gender=='F'and (state=='Others' or state=='Sikkim' or state=='Dadra and Nagar Haveli') group by district order by sum(aadhaar_generated)/(sum(aadhaar_generated)+sum(rejected))*100 DESC LIMIT 3").show()

21

create table if not exists aadhaar_details_staging(registrar string,private_agency string,state string,district string,sub_district string,pincode string,gender string,age int,aadhaar_generated int,rejected int,provide_email int,provide_mobile int)
clustered by (age) into 10 buckets
row format delimited fields terminated by ‘,’
stored as textfile

TBLPROPERTIES(‘serialization.null,format’=’’,’skip.header.line.count’=’1’;

```
select round(sum(aadhaar_generated)/(sum(aadhaar_generated)+sum(rejected))*100,2)
from aadhaar_details_staging;
```