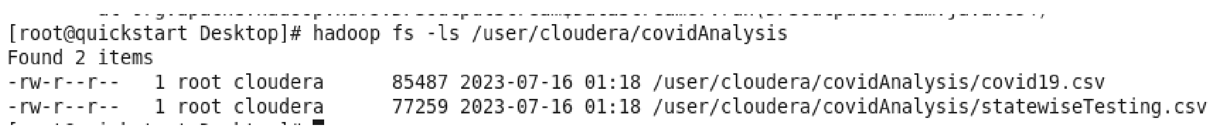
Step 1. Data Preprocessing

1.1 Create directories in hdfs to store the csv files

hadoop fs -mkdir /user/cloudera/covidAnalysis

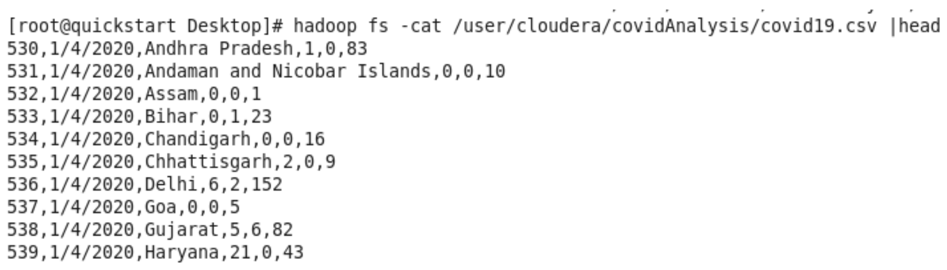
1.2 Copy the files from local to hdfs

hadoop fs -copyFromLocal covid/covid19.csv /user/cloudera/covidAnalysis
hadoop fs -copyFromLocal covid/statewiseTesting.csv /user/cloudera/covidAnalysis

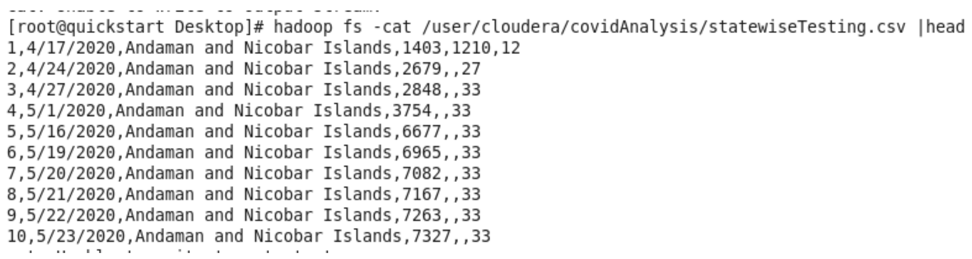


```
[root@quickstart Desktop]# hadoop fs -ls /user/cloudera/covidAnalysis
Found 2 items
-rw-r--r--   1 root cloudera      85487 2023-07-16 01:18 /user/cloudera/covidAnalysis/covid19.csv
-rw-r--r--   1 root cloudera      77259 2023-07-16 01:18 /user/cloudera/covidAnalysis/statewiseTesting.csv
```

1.3 Visualize the content of the files in hdfs

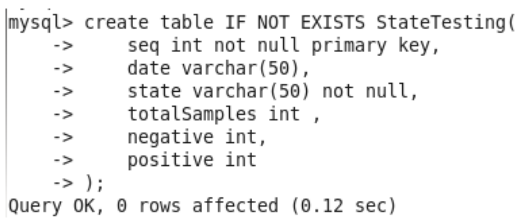


```
[root@quickstart Desktop]# hadoop fs -cat /user/cloudera/covidAnalysis/covid19.csv |head
530,1/4/2020,Andhra Pradesh,1,0,83
531,1/4/2020,Andaman and Nicobar Islands,0,0,10
532,1/4/2020,Assam,0,0,1
533,1/4/2020,Bihar,0,1,23
534,1/4/2020,Chandigarh,0,0,16
535,1/4/2020,Chhattisgarh,2,0,9
536,1/4/2020,Delhi,6,2,152
537,1/4/2020,Goa,0,0,5
538,1/4/2020,Gujarat,5,6,82
539,1/4/2020,Haryana,21,0,43
```

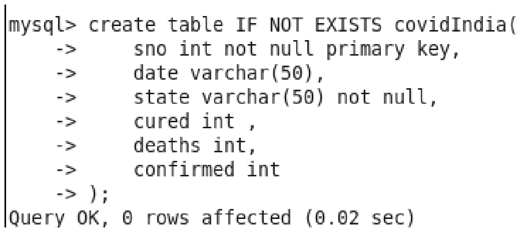


```
[root@quickstart Desktop]# hadoop fs -cat /user/cloudera/covidAnalysis/statewiseTesting.csv |head
1,4/17/2020,Andaman and Nicobar Islands,1403,1210,12
2,4/24/2020,Andaman and Nicobar Islands,2679,,27
3,4/27/2020,Andaman and Nicobar Islands,2848,,33
4,5/1/2020,Andaman and Nicobar Islands,3754,,33
5,5/16/2020,Andaman and Nicobar Islands,6677,,33
6,5/19/2020,Andaman and Nicobar Islands,6965,,33
7,5/20/2020,Andaman and Nicobar Islands,7082,,33
8,5/21/2020,Andaman and Nicobar Islands,7167,,33
9,5/22/2020,Andaman and Nicobar Islands,7263,,33
10,5/23/2020,Andaman and Nicobar Islands,7327,,33
```

1.4 Creation of Tables in Mysql



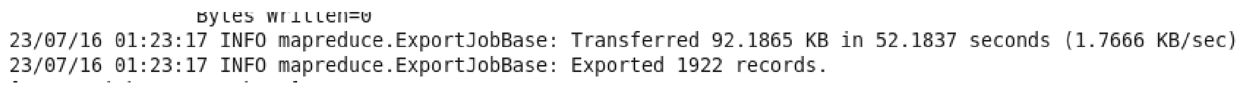
```
mysql> create table IF NOT EXISTS StateTesting(
    ->     seq int not null primary key,
    ->     date varchar(50),
    ->     state varchar(50) not null,
    ->     totalSamples int ,
    ->     negative int,
    ->     positive int
    -> );
Query OK, 0 rows affected (0.12 sec)
```



```
mysql> create table IF NOT EXISTS covidIndia(
    ->     sno int not null primary key,
    ->     date varchar(50),
    ->     state varchar(50) not null,
    ->     cured int ,
    ->     deaths int,
    ->     confirmed int
    -> );
Query OK, 0 rows affected (0.02 sec)
```

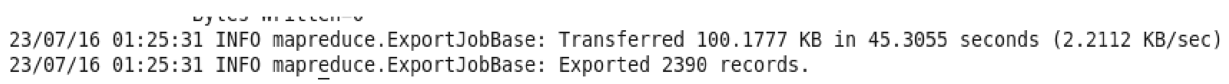
1.5 Sqoop Export of data from hdfs to Mysql

```
sqoop export \
--connect jdbc:mysql://quickstart.cloudera:3306/test_db \
--username root \
--password cloudera \
--table StateTesting \
--lines-terminated-by ',' \
--export-dir /user/cloudera/covidAnalysis/statewiseTesting.csv
```



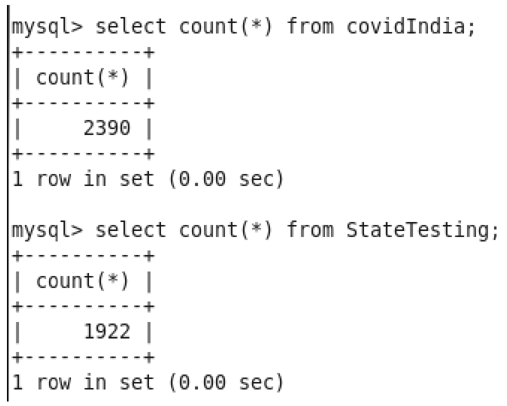
```
                Bytes Written=0
23/07/16 01:23:17 INFO mapreduce.ExportJobBase: Transferred 92.1865 KB in 52.1837 seconds (1.7666 KB/sec)
23/07/16 01:23:17 INFO mapreduce.ExportJobBase: Exported 1922 records.
```

```
sqoop export \
--connect jdbc:mysql://quickstart.cloudera:3306/test_db \
--username root \
--password cloudera \
--table covidIndia\
--lines-terminated-by ',' \
--export-dir /user/cloudera/covidAnalysis/covid19.csv
```



```
                Bytes Written=0
23/07/16 01:25:31 INFO mapreduce.ExportJobBase: Transferred 100.1777 KB in 45.3055 seconds (2.2112 KB/sec)
23/07/16 01:25:31 INFO mapreduce.ExportJobBase: Exported 2390 records.
```

1.6 Verify in Mysql



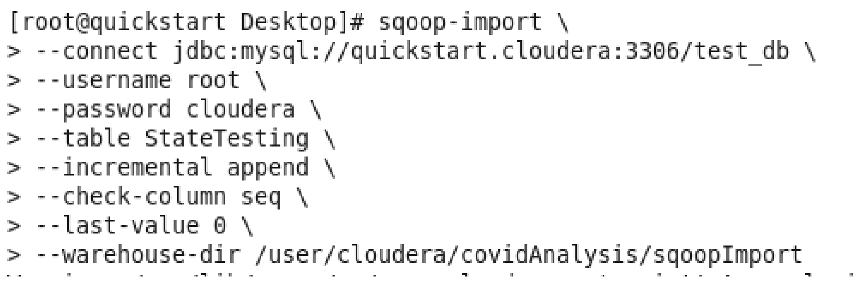
```
mysql> select count(*) from covidIndia;
+----------+
| count(*) |
+----------+
|     2390 |
+----------+
1 row in set (0.00 sec)

mysql> select count(*) from StateTesting;
+----------+
| count(*) |
+----------+
|     1922 |
+----------+
1 row in set (0.00 sec)
```

1.7 Delete data from hdfs

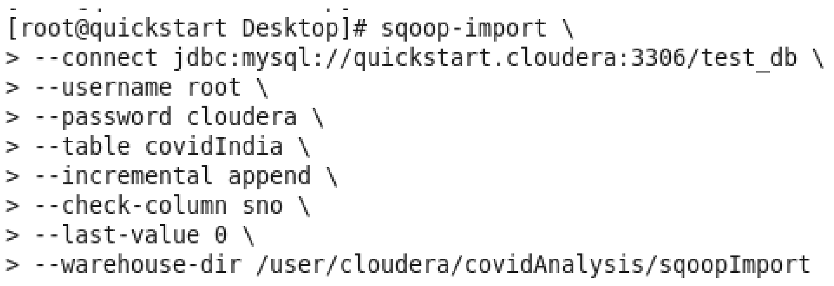
Step 2- Sqoop Import

Command:



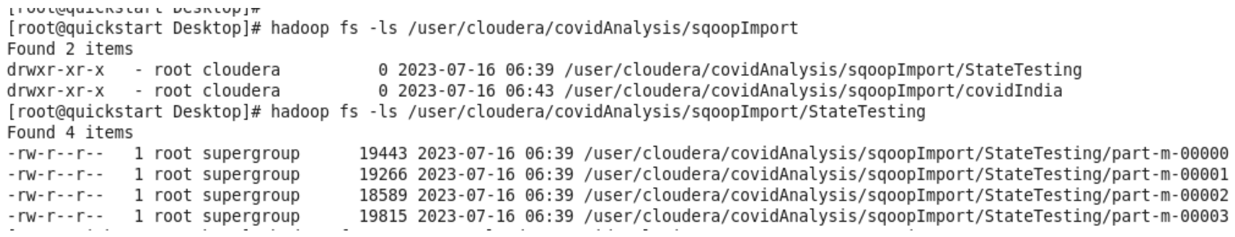
```
[root@quickstart Desktop]# sqoop-import \
> --connect jdbc:mysql://quickstart.cloudera:3306/test_db \
> --username root \
> --password cloudera \
> --table StateTesting \
> --incremental append \
> --check-column seq \
> --last-value 0 \
> --warehouse-dir /user/cloudera/covidAnalysis/sqoopImport
```

Command:



```
[root@quickstart Desktop]# sqoop-import \
> --connect jdbc:mysql://quickstart.cloudera:3306/test_db \
> --username root \
> --password cloudera \
> --table covidIndia \
> --incremental append \
> --check-column sno \
> --last-value 0 \
> --warehouse-dir /user/cloudera/covidAnalysis/sqoopImport
```

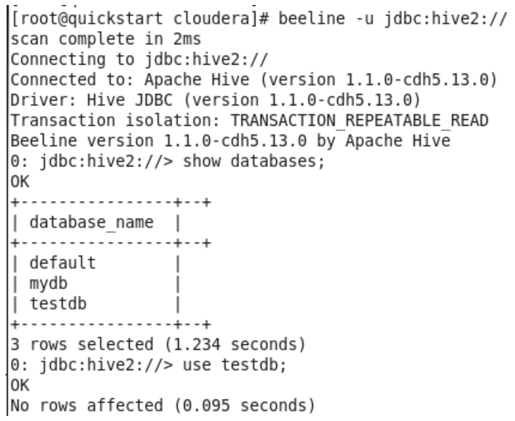
Output:



```
[root@quickstart Desktop]# hadoop fs -ls /user/cloudera/covidAnalysis/sqoopImport
Found 2 items
drwxr-xr-x   - root cloudera          0 2023-07-16 06:39 /user/cloudera/covidAnalysis/sqoopImport/StateTesting
drwxr-xr-x   - root cloudera          0 2023-07-16 06:43 /user/cloudera/covidAnalysis/sqoopImport/covidIndia
[root@quickstart Desktop]# hadoop fs -ls /user/cloudera/covidAnalysis/sqoopImport/StateTesting
Found 4 items
-rw-r--r--   1 root supergroup      19443 2023-07-16 06:39 /user/cloudera/covidAnalysis/sqoopImport/StateTesting/part-m-00000
-rw-r--r--   1 root supergroup      19266 2023-07-16 06:39 /user/cloudera/covidAnalysis/sqoopImport/StateTesting/part-m-00001
-rw-r--r--   1 root supergroup      18589 2023-07-16 06:39 /user/cloudera/covidAnalysis/sqoopImport/StateTesting/part-m-00002
-rw-r--r--   1 root supergroup      19815 2023-07-16 06:39 /user/cloudera/covidAnalysis/sqoopImport/StateTesting/part-m-00003
```

Step 3- Create Hive External Tables on top of data in HDFS

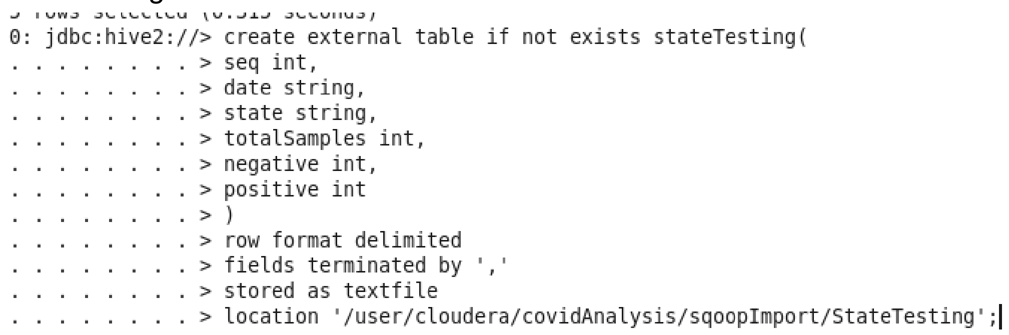
3.1 Connect to hive and select/create a database



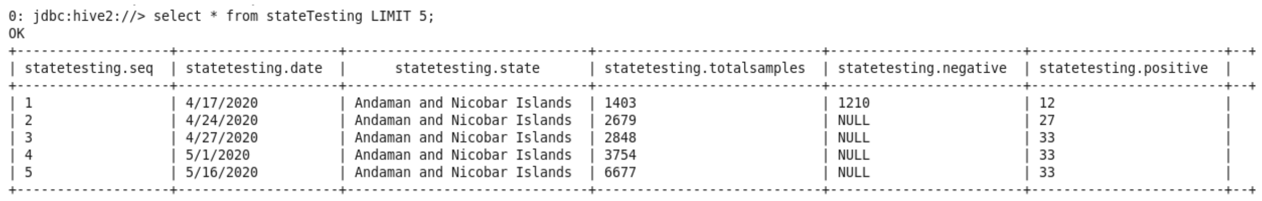
```
[root@quickstart cloudera]# beeline -u jdbc:hive2://
scan complete in 2ms
Connecting to jdbc:hive2://
Connected to: Apache Hive (version 1.1.0-cdh5.13.0)
Driver: Hive JDBC (version 1.1.0-cdh5.13.0)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 1.1.0-cdh5.13.0 by Apache Hive
0: jdbc:hive2://> show databases;
OK
+----------------+--+
| database_name  |
+----------------+--+
| default        |
| mydb           |
| testdb         |
+----------------+--+
3 rows selected (1.234 seconds)
0: jdbc:hive2://> use testdb;
OK
No rows affected (0.095 seconds)
```

3.2 Creation of Hive External Tables on top of data in HDFS

StateTesting table



```
5 rows selected (0.315 seconds)
0: jdbc:hive2://> create external table if not exists stateTesting(
. . . . . . . . > seq int,
. . . . . . . . > date string,
. . . . . . . . > state string,
. . . . . . . . > totalSamples int,
. . . . . . . . > negative int,
. . . . . . . . > positive int
. . . . . . . . > )
. . . . . . . . > row format delimited
. . . . . . . . > fields terminated by ','
. . . . . . . . > stored as textfile
. . . . . . . . > location '/user/cloudera/covidAnalysis/sqoopImport/StateTesting';
```



```
0: jdbc:hive2://> select * from stateTesting LIMIT 5;
OK
+-----------------+--------------------+-----------------------------+--------------------------+-----------------------+--------------------------+--+
| statetesting.seq  | statetesting.date  |      statetesting.state      | statetesting.totalsamples  | statetesting.negative  | statetesting.positive  |
+-----------------+--------------------+-----------------------------+--------------------------+-----------------------+--------------------------+--+
| 1                 | 4/17/2020          | Andaman and Nicobar Islands  | 1403                       | 1210                   | 12                     |
| 2                 | 4/24/2020          | Andaman and Nicobar Islands  | 2679                       | NULL                   | 27                     |
| 3                 | 4/27/2020          | Andaman and Nicobar Islands  | 2848                       | NULL                   | 33                     |
| 4                 | 5/1/2020           | Andaman and Nicobar Islands  | 3754                       | NULL                   | 33                     |
| 5                 | 5/16/2020          | Andaman and Nicobar Islands  | 6677                       | NULL                   | 33                     |
+-----------------+--------------------+-----------------------------+--------------------------+-----------------------+--------------------------+--+
```

Similarly, we do for covidIndia Table

Step 4- Create Optimized External tables in Hive

Optimizations Applied-

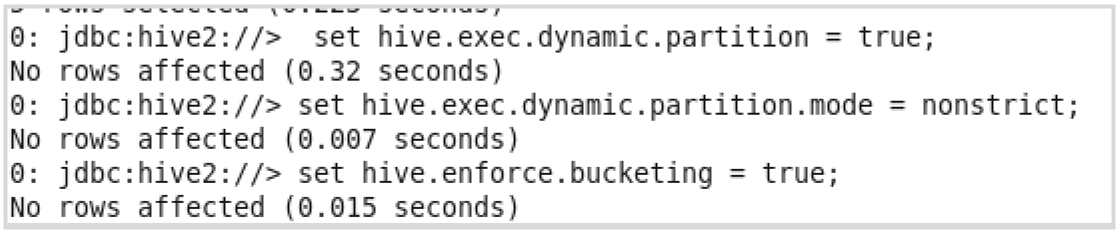
- File format used- ORC for Quicker and Efficient Reads
- Compression Codec used- Snappy for Fast Compression
- Partitioning on State Column
- Bucketing on Date Column

4.1 Create directories in hdfs for the Dynamically created Partitions:

hadoop fs -mkdir /user/cloudera/covidAnalysis/partitions_testing
hadoop fs -mkdir /user/cloudera/covidAnalysis/partitions_covidIndia

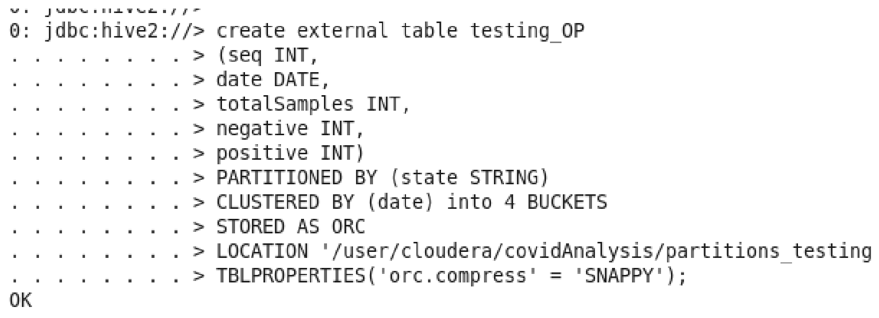
4.2 Enabling Dynamic Partitioning and Bucketing in Hive:

set hive.exec.dynamic.partition = true;
set hive.exec.dynamic.partition.mode = nonstrict;
set hive.enforce.bucketing = true;

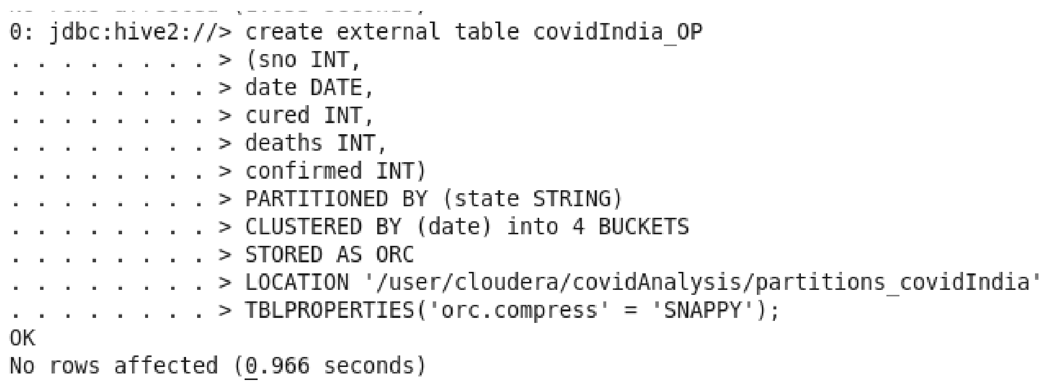


```
0: jdbc:hive2://>  set hive.exec.dynamic.partition = true;
No rows affected (0.32 seconds)
0: jdbc:hive2://> set hive.exec.dynamic.partition.mode = nonstrict;
No rows affected (0.007 seconds)
0: jdbc:hive2://> set hive.enforce.bucketing = true;
No rows affected (0.015 seconds)
```

4.3 Optimized External tables creation in Hive :



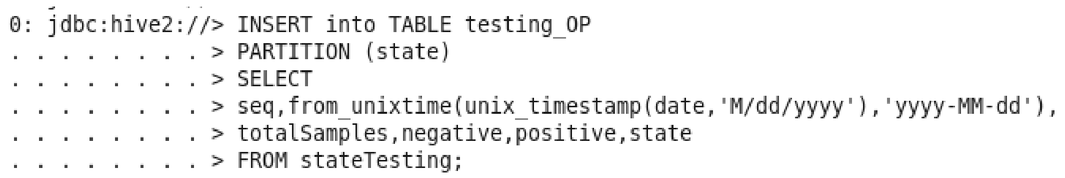
```
0: jdbc:hive2://> create external table testing_OP
. . . . . . . . > (seq INT,
. . . . . . . . > date DATE,
. . . . . . . . > totalSamples INT,
. . . . . . . . > negative INT,
. . . . . . . . > positive INT)
. . . . . . . . > PARTITIONED BY (state STRING)
. . . . . . . . > CLUSTERED BY (date) into 4 BUCKETS
. . . . . . . . > STORED AS ORC
. . . . . . . . > LOCATION '/user/cloudera/covidAnalysis/partitions_testing
. . . . . . . . > TBLPROPERTIES('orc.compress' = 'SNAPPY');
OK
```



```
0: jdbc:hive2://> create external table covidIndia_OP
. . . . . . . . > (sno INT,
. . . . . . . . > date DATE,
. . . . . . . . > cured INT,
. . . . . . . . > deaths INT,
. . . . . . . . > confirmed INT)
. . . . . . . . > PARTITIONED BY (state STRING)
. . . . . . . . > CLUSTERED BY (date) into 4 BUCKETS
. . . . . . . . > STORED AS ORC
. . . . . . . . > LOCATION '/user/cloudera/covidAnalysis/partitions_covidIndia'
. . . . . . . . > TBLPROPERTIES('orc.compress' = 'SNAPPY');
OK
No rows affected (0.966 seconds)
```

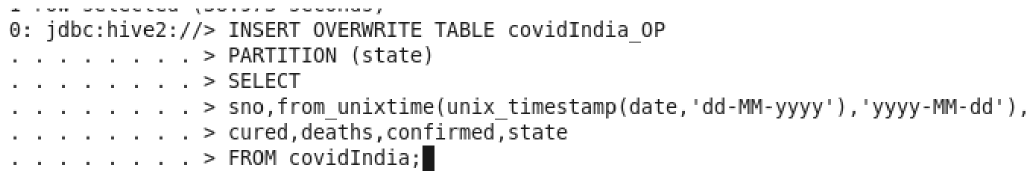
Step 5: Load data to the optimized hive tables from normal hive tables.

The date in stateTesting table is of the format M/dd/yyyy . We need to convert it to yyyy-MM-dd format.



```
0: jdbc:hive2://> INSERT into TABLE testing_OP
. . . . . . . . > PARTITION (state)
. . . . . . . . > SELECT
. . . . . . . . > seq,from_unixtime(unix_timestamp(date,'M/dd/yyyy'),'yyyy-MM-dd'),
. . . . . . . . > totalSamples,negative,positive,state
. . . . . . . . > FROM stateTesting;
```

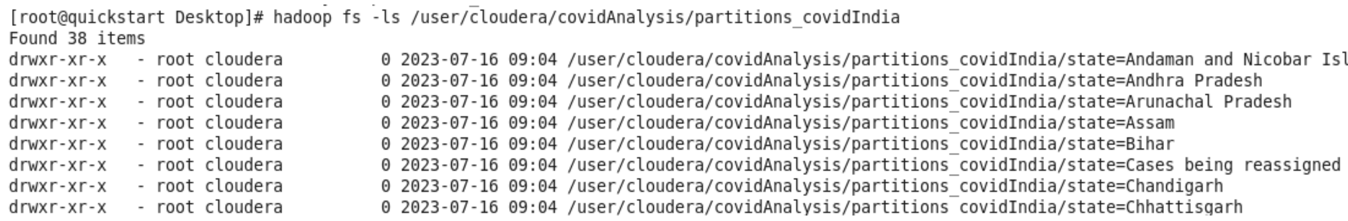
The date in covidIndia table is of the format dd-MM-yyyy . We need to convert it to yyyy-MM-dd format.



```
0: jdbc:hive2://> INSERT OVERWRITE TABLE covidIndia_OP
. . . . . . . . > PARTITION (state)
. . . . . . . . > SELECT
. . . . . . . . > sno,from_unixtime(unix_timestamp(date,'dd-MM-yyyy'),'yyyy-MM-dd'),
. . . . . . . . > cured,deaths,confirmed,state
. . . . . . . . > FROM covidIndia;
```

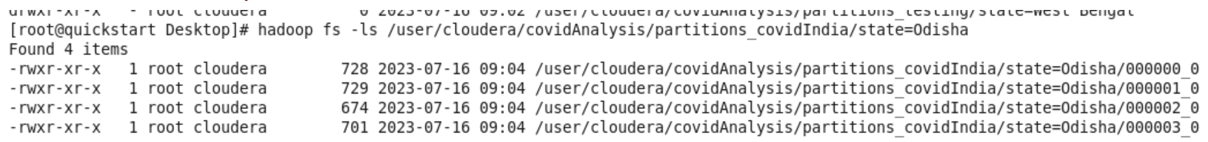
Verification:

Partitions based on state



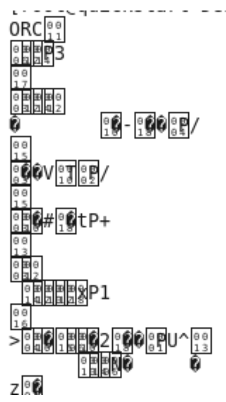
```
[root@quickstart Desktop]# hadoop fs -ls /user/cloudera/covidAnalysis/partitions_covidIndia
Found 38 items
drwxr-xr-x   - root cloudera          0 2023-07-16 09:04 /user/cloudera/covidAnalysis/partitions_covidIndia/state=Andaman and Nicobar Isl
drwxr-xr-x   - root cloudera          0 2023-07-16 09:04 /user/cloudera/covidAnalysis/partitions_covidIndia/state=Andhra Pradesh
drwxr-xr-x   - root cloudera          0 2023-07-16 09:04 /user/cloudera/covidAnalysis/partitions_covidIndia/state=Arunachal Pradesh
drwxr-xr-x   - root cloudera          0 2023-07-16 09:04 /user/cloudera/covidAnalysis/partitions_covidIndia/state=Assam
drwxr-xr-x   - root cloudera          0 2023-07-16 09:04 /user/cloudera/covidAnalysis/partitions_covidIndia/state=Bihar
drwxr-xr-x   - root cloudera          0 2023-07-16 09:04 /user/cloudera/covidAnalysis/partitions_covidIndia/state=Cases being reassigned
drwxr-xr-x   - root cloudera          0 2023-07-16 09:04 /user/cloudera/covidAnalysis/partitions_covidIndia/state=Chandigarh
drwxr-xr-x   - root cloudera          0 2023-07-16 09:04 /user/cloudera/covidAnalysis/partitions_covidIndia/state=Chhattisgarh
```

4 Buckets in each partition



```
[root@quickstart Desktop]# hadoop fs -ls /user/cloudera/covidAnalysis/partitions_covidIndia/state=Odisha
Found 4 items
-rwxr-xr-x   1 root cloudera        728 2023-07-16 09:04 /user/cloudera/covidAnalysis/partitions_covidIndia/state=Odisha/000000_0
-rwxr-xr-x   1 root cloudera        729 2023-07-16 09:04 /user/cloudera/covidAnalysis/partitions_covidIndia/state=Odisha/000001_0
-rwxr-xr-x   1 root cloudera        674 2023-07-16 09:04 /user/cloudera/covidAnalysis/partitions_covidIndia/state=Odisha/000002_0
-rwxr-xr-x   1 root cloudera        701 2023-07-16 09:04 /user/cloudera/covidAnalysis/partitions_covidIndia/state=Odisha/000003_0
```

Data stored in ORC format



```
ORC
```

Step 6-Inner Join two tables in Hive and get a consolidated table.

Performing Map side join on two columns ‘date’ and ‘state’.
Here it is assumed that the State_Testing table is small enough to fit in memory

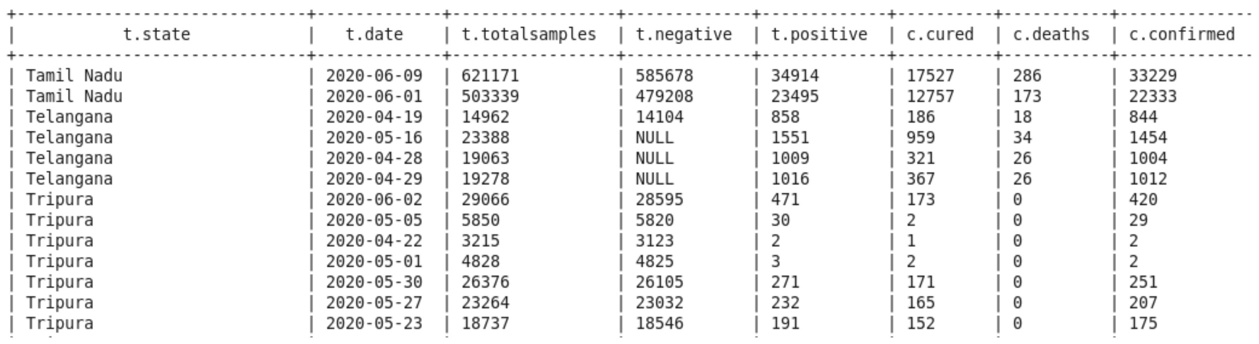
6.1 Set the below Properties

set hive.auto.convert.join = false;
set hive.ignore.mapjoin.hint = false;

6.2 Execute Inner Join as Mapside join (testing_OP is the smaller table)

SELECT /+ MAPJOIN(T)*
**/T.state,T.date,T.totalSamples,T.negative,T.positive,C.cured,C.deaths,*
C.confirmed FROM testing_OP T JOIN covidIndia_OP C
ON (C.state = T.state) AND (C.date = T.date);

Joined Table Snapshot

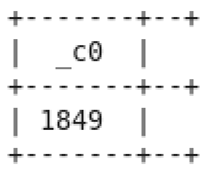


```
+-----------------------------+------------+----------------+------------+------------+----------+-----------+------------
|           t.state           |   t.date   | t.totalsamples | t.negative | t.positive | c.cured  | c.deaths  | c.confirmed
+-----------------------------+------------+----------------+------------+------------+----------+-----------+------------
| Tamil Nadu                  | 2020-06-09 | 621171         | 585678     | 34914      | 17527    | 286       | 33229
| Tamil Nadu                  | 2020-06-01 | 503339         | 479208     | 23495      | 12757    | 173       | 22333
| Telangana                   | 2020-04-19 | 14962          | 14104      | 858        | 186      | 18        | 844
| Telangana                   | 2020-05-16 | 23388          | NULL       | 1551       | 959      | 34        | 1454
| Telangana                   | 2020-04-28 | 19063          | NULL       | 1009       | 321      | 26        | 1004
| Telangana                   | 2020-04-29 | 19278          | NULL       | 1016       | 367      | 26        | 1012
| Tripura                     | 2020-06-02 | 29066          | 28595      | 471        | 173      | 0         | 420
| Tripura                     | 2020-05-05 | 5850           | 5820       | 30         | 2        | 0         | 29
| Tripura                     | 2020-04-22 | 3215           | 3123       | 2          | 1        | 0         | 2
| Tripura                     | 2020-05-01 | 4828           | 4825       | 3          | 2        | 0         | 2
| Tripura                     | 2020-05-30 | 26376          | 26105      | 271        | 171      | 0         | 251
| Tripura                     | 2020-05-27 | 23264          | 23032      | 232        | 165      | 0         | 207
| Tripura                     | 2020-05-23 | 18737          | 18546      | 191        | 152      | 0         | 175
```

6.3 Create a consolidated table after join

```
CREATE TABLE covid_details AS
SELECT /*+ MAPJOIN(T)
*/T.state,T.date,T.totalSamples,T.negative,T.positive,C.cured,C.deaths,
C.confirmed FROM testing_OP T JOIN covidIndia_OP C
ON (C.state = T.state) AND (C.date = T.date);
```

Count:



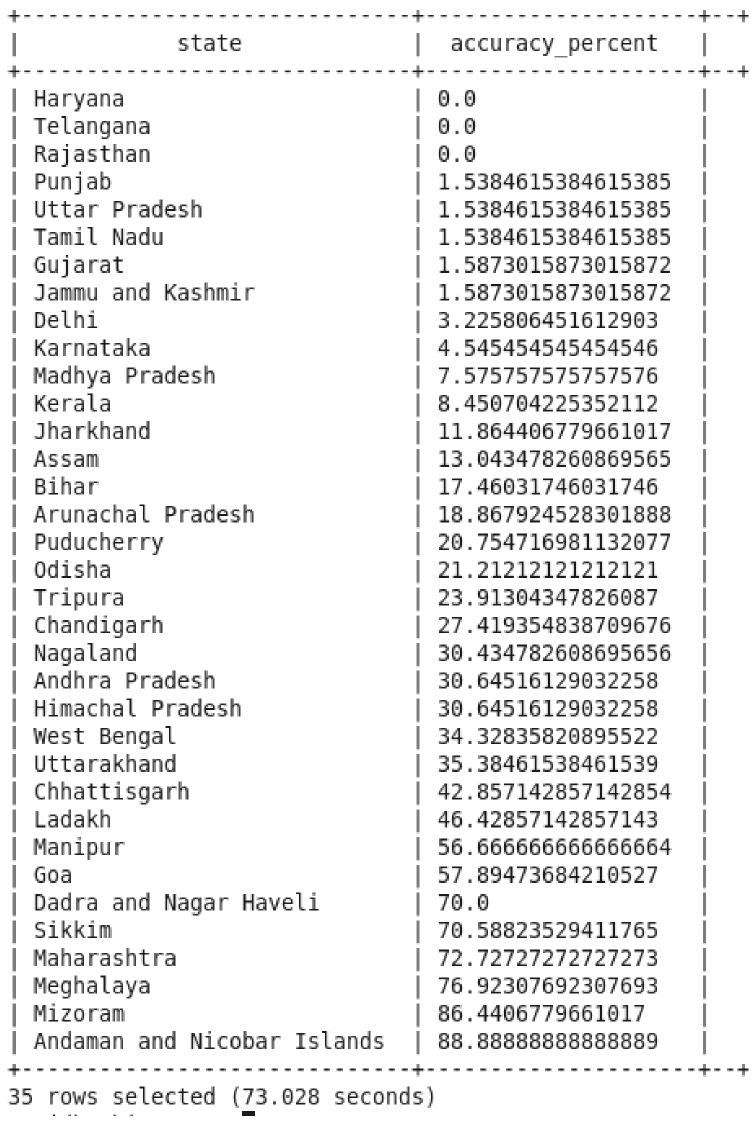
```
+-------+--+
|  _c0  |
+-------+--+
| 1849  |
+-------+--+
```

Step 7- Analysis

1. **Ideally, the number of samples tested positive and number of covid cases confirmed must be the same. See which state/states have more consistent data collection like The number of positive samples (table1) match mostly with number of confirmed cases(table2), for which state.**

   Ideally, the number of samples tested positive and number of covid cases confirmed must be the same. So we find the percentage of records in each state where the data is consistent.

```
select state,avg(case when positive = confirmed then 1 else 0 end)*100 as
accuracy_percent
from covid_details
group by state
order by accuracy_percent;
```



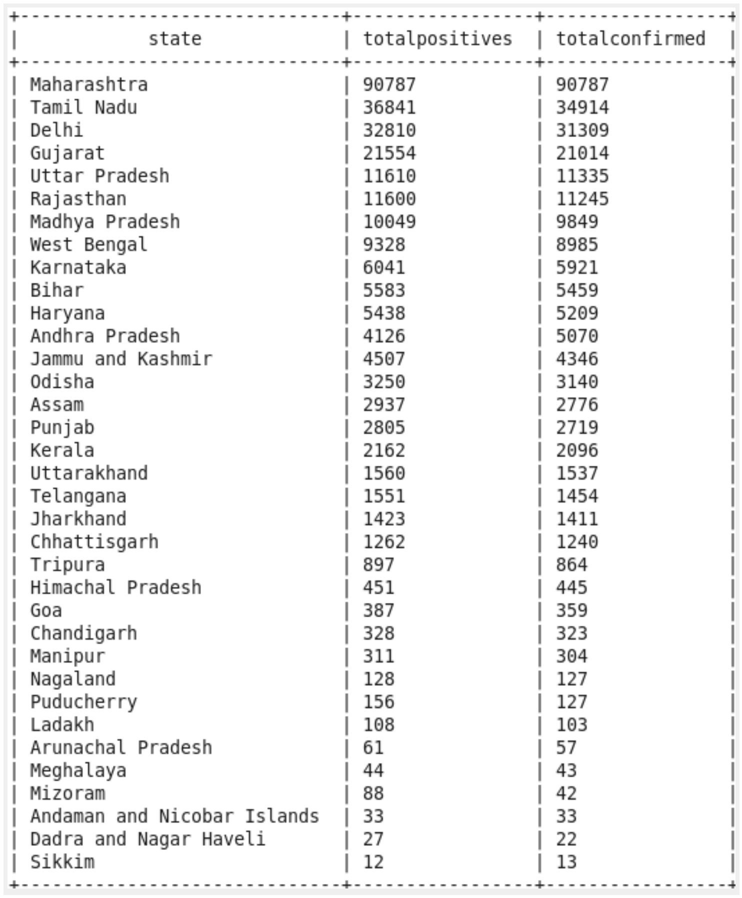
```
+------------------------------+---------------------+--+
|            state             |  accuracy_percent   |
+------------------------------+---------------------+--+
| Haryana                      | 0.0                 |
| Telangana                    | 0.0                 |
| Rajasthan                    | 0.0                 |
| Punjab                       | 1.5384615384615385  |
| Uttar Pradesh                | 1.5384615384615385  |
| Tamil Nadu                   | 1.5384615384615385  |
| Gujarat                      | 1.5873015873015872  |
| Jammu and Kashmir            | 1.5873015873015872  |
| Delhi                        | 3.225806451612903   |
| Karnataka                    | 4.545454545454546   |
| Madhya Pradesh               | 7.575757575757576   |
| Kerala                       | 8.450704225352112   |
| Jharkhand                    | 11.864406779661017  |
| Assam                        | 13.043478260869565  |
| Bihar                        | 17.46031746031746   |
| Arunachal Pradesh            | 18.867924528301888  |
| Puducherry                   | 20.754716981132077  |
| Odisha                       | 21.21212121212121   |
| Tripura                      | 23.91304347826087   |
| Chandigarh                   | 27.419354838709676  |
| Nagaland                     | 30.434782608695656  |
| Andhra Pradesh               | 30.64516129032258   |
| Himachal Pradesh             | 30.64516129032258   |
| West Bengal                  | 34.32835820895522   |
| Uttarakhand                  | 35.38461538461539   |
| Chhattisgarh                 | 42.857142857142854  |
| Ladakh                       | 46.42857142857143   |
| Manipur                      | 56.666666666666664  |
| Goa                          | 57.89473684210527   |
| Dadra and Nagar Haveli       | 70.0                |
| Sikkim                       | 70.58823529411765   |
| Maharashtra                  | 72.72727272727273   |
| Meghalaya                    | 76.92307692307693   |
| Mizoram                      | 86.4406779661017    |
| Andaman and Nicobar Islands  | 88.88888888888889   |
+------------------------------+---------------------+--+
35 rows selected (73.028 seconds)
```

According to the above results, we can conclude the top 5 states with accurate covid testing for the given 2 months are **- Sikkim, Maharashtra, Meghalaya, Mizoram and Andaman and Nicobar Islands**
While the worst-performing ones are - **Haryana, Telangana, Rajasthan, Punjab, Uttar Pradesh, Tamil Nadu and Gujarat**

2. **For every state, find the total number of confirmed cases reported and also the total number of positive samples tested, in the entire duration of 2 months, starting with the state with the highest cases.**

If we scan the data we can see the data is cumulative for example TotalSamples shows cumulative total, Confirmed field shows cumulative value. So the value in each day is actually the total number till that day. Hence we only need to find the maximum number.

```
select state,max(positive) as totalPositives , max(confirmed) as totalConfirmed
from covid_details
group by state
order by totalConfirmed desc;
```



```
+------------------------------+----------------+-----------------+
|            state             | totalpositives | totalconfirmed  |
+------------------------------+----------------+-----------------+
| Maharashtra                  | 90787          | 90787           |
| Tamil Nadu                   | 36841          | 34914           |
| Delhi                        | 32810          | 31309           |
| Gujarat                      | 21554          | 21014           |
| Uttar Pradesh                | 11610          | 11335           |
| Rajasthan                    | 11600          | 11245           |
| Madhya Pradesh               | 10049          | 9849            |
| West Bengal                  | 9328           | 8985            |
| Karnataka                    | 6041           | 5921            |
| Bihar                        | 5583           | 5459            |
| Haryana                      | 5438           | 5209            |
| Andhra Pradesh               | 4126           | 5070            |
| Jammu and Kashmir            | 4507           | 4346            |
| Odisha                       | 3250           | 3140            |
| Assam                        | 2937           | 2776            |
| Punjab                       | 2805           | 2719            |
| Kerala                       | 2162           | 2096            |
| Uttarakhand                  | 1560           | 1537            |
| Telangana                    | 1551           | 1454            |
| Jharkhand                    | 1423           | 1411            |
| Chhattisgarh                 | 1262           | 1240            |
| Tripura                      | 897            | 864             |
| Himachal Pradesh             | 451            | 445             |
| Goa                          | 387            | 359             |
| Chandigarh                   | 328            | 323             |
| Manipur                      | 311            | 304             |
| Nagaland                     | 128            | 127             |
| Puducherry                   | 156            | 127             |
| Ladakh                       | 108            | 103             |
| Arunachal Pradesh            | 61             | 57              |
| Meghalaya                    | 44             | 43              |
| Mizoram                      | 88             | 42              |
| Andaman and Nicobar Islands  | 33             | 33              |
| Dadra and Nagar Haveli       | 27             | 22              |
| Sikkim                       | 12             | 13              |
+------------------------------+----------------+-----------------+
```