Analyzing Aadhar Cards Data

edureka!

edureka!

Version 2.0

Problem statement:

1. Find out the total number of cards approved by States.
2. Find out the total number of cards rejected by states.
3. Find out the total number of cards approved by cities.
4. Find out the total number of cards rejected by cities.

Important Links: Link For All codes:

https://edureka.wistia.com/medias/tm1r5cndt2/download?media_file_id=66797296

Dataset:

https://edureka.wistia.com/medias/p591t1geg8/download?media_file_id=66797131

Data Set Description:

The data set consists of the following fields.

State: This field consists of the state names from all over India

City: This field consists of city names in all states

Approved: This fields consists of the total count of approved cards in numbers

Rejected: This field consists of the total count of rejected cards in numbers

Codes and Explanation:

First we need to create a directory in HDFS. Creating a directory called **edureka_project** in hdfs.

```
[edureka@localhost ~]$ hadoop dfs -mkdir /edureka_project
```

Usecase1:

In this use case we are finding the total number of cards approved by States.

```
Usecase1:

    1.  Find out the total number of cards approved by States.

cards = load '/edureka_project/Dataset/aadhar' as
(state:chararray,city:chararray,approved:int,rejected:int);

state_grouped = group cards by state;

state_count = foreach state_grouped generate group, SUM(cards.approved);

store state_count into '/edureka_project/stateapproved';
```

Explanation for usecase1:

- Load Aadhar details
- group them by states
- summing up the values of cards approved by each state
- Finally storing the output into HDFS.

Usecase1 Output:

Below is the sample output screen for usecase1

```
Goa     30
Assam 832
Bihar 162152
Delhi 3303
Kerala      4092
Odisha      46612
Others      33
Punjab      5092
Sikkim      17
Gujarat     31178
Haryana     12972
```

Usecase2:

In this use case we are finding total number of cards rejected by each states.

```
Usecase2:

    2.  Find out the total number of cards rejected by states.

cards = load '/edureka_project/Dataset/aadhar' as
(state:chararray,city:chararray,approved:int,rejected:int);

state_grouped = group cards by state;

state_count = foreach state_grouped generate group, SUM(cards.rejected);

store state_count into '/edureka_project/staterejected';
```

Explanation for usecase2:

- Load Aadhar details
- group them by states
- summing up the values of cards rejected by each state

- Finally storing the output into HDFS.

Usecase2 Output:

```
Goa     7
Assam 3
Bihar 10521
Delhi 512
Kerala      187
Odisha      2788
Others      660
Punjab      572
Sikkim      0
Gujarat     877
Haryana     2577
Manipur     12
```

Usecase3:

In this use case we are finding the total number of cards approved by cities.

Usecase3:

   3. Find out the total number of cards approved by cities.

```
cards2 = load '/edureka_project/Dataset/aadhar' as (state:chararray, city:chararray, approved:int, rejected:int);

city_grouped = group cards2 by city;

city_count = foreach city_grouped generate group, SUM(cards2.approved);

store city_count into '/edureka_project/cityapproved';
```

Explanation for usecase3:

- Load Aadhar details
- group them by states
- summing up the values of cards approved
  by cities

- Finally storing the output into HDFS.

Usecase3 Output:

```
Diu    42
Leh    1
Mau    8204
Una    3
Agra   4458
Beed   1526
Dhar   487
Doda   93
Durg   474
Etah   3990
Gaya   10977
```

Usecase 4:

In this use case we are finding the total number of cards rejected by cities.

Usecase4:

4. Find out the total number of cards rejected by cities.

cards2 = load '/edureka_project/Dataset/aadhar' as (state:chararray, city:chararray, approved:int, rejected:int);

city_grouped = group cards2 by city;

city_count = foreach city_grouped generate group, SUM(cards2.rejected);

store city_count into '/edureka_project/cityrejected';

Explanation for usecase4:

- Load Aadhar details
- Group them by states
- Summing up the values of cards rejected by cities

- Finally storing the output into HDFS.

Usecase4 Output:

```
Diu    2
Leh    0
Mau    213
Una    0
Agra   492
Beed   40
Dhar   19
Doda   0
Durg   95
Etah   167
Gaya   738
```