BIG DATA LABORATORY

Course Code: ISL75 **Credit: 0:0:1**
Prerequisite: Nil **Contact Hours: 14L**

Course Coordinator: Dr. Rajeshwari S B

PART-A

1. Compile all java files (driver.java mapper.java reducer.java)

   javac -d . *.java
2. Set driver class in manifest
   echo Main-Class: weather.driver > Manifest.txt
3. Create an executable jar file
   jar cfm weather.jar Manifest.txt weather/*.class
4. input.txt is input file for Weather create Input File
5. Run the jar file
   hadoop jar weather.jar input.txt output
6. To see the Output:
   cat output/*

Write MapReduce programs for the following using Java:

1. To count the number of occurrences of each word in a given input text.

<Goodbye, 1>
<Hadoop, 2>
<Hello, 1>

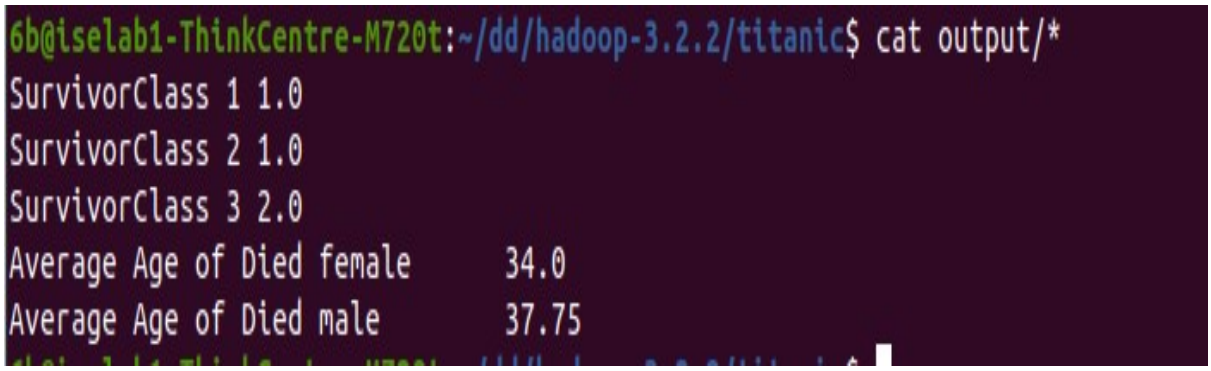
2. To read N natural numbers and display the sum along with Odd and Even count.

```
ibmlab@ibmlab: ~/1MS23SCS17/hadoop-3.2.2/Programs/oddeven
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/oddeven$ ls
driver.java  input.txt  Manifest.txt  mapper.java  oddeven  oddeven.jar  output  reducer.java
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/oddeven$
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/oddeven$
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/oddeven$
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/oddeven$
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/oddeven$
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/oddeven$ cat output/*
Sum of even Numbers     30
even Number count       5
Sum of odd Numbers      25
odd Number count        5
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/oddeven$
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/oddeven$
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/oddeven$
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/oddeven$
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/oddeven$
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/oddeven$
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/oddeven$
```

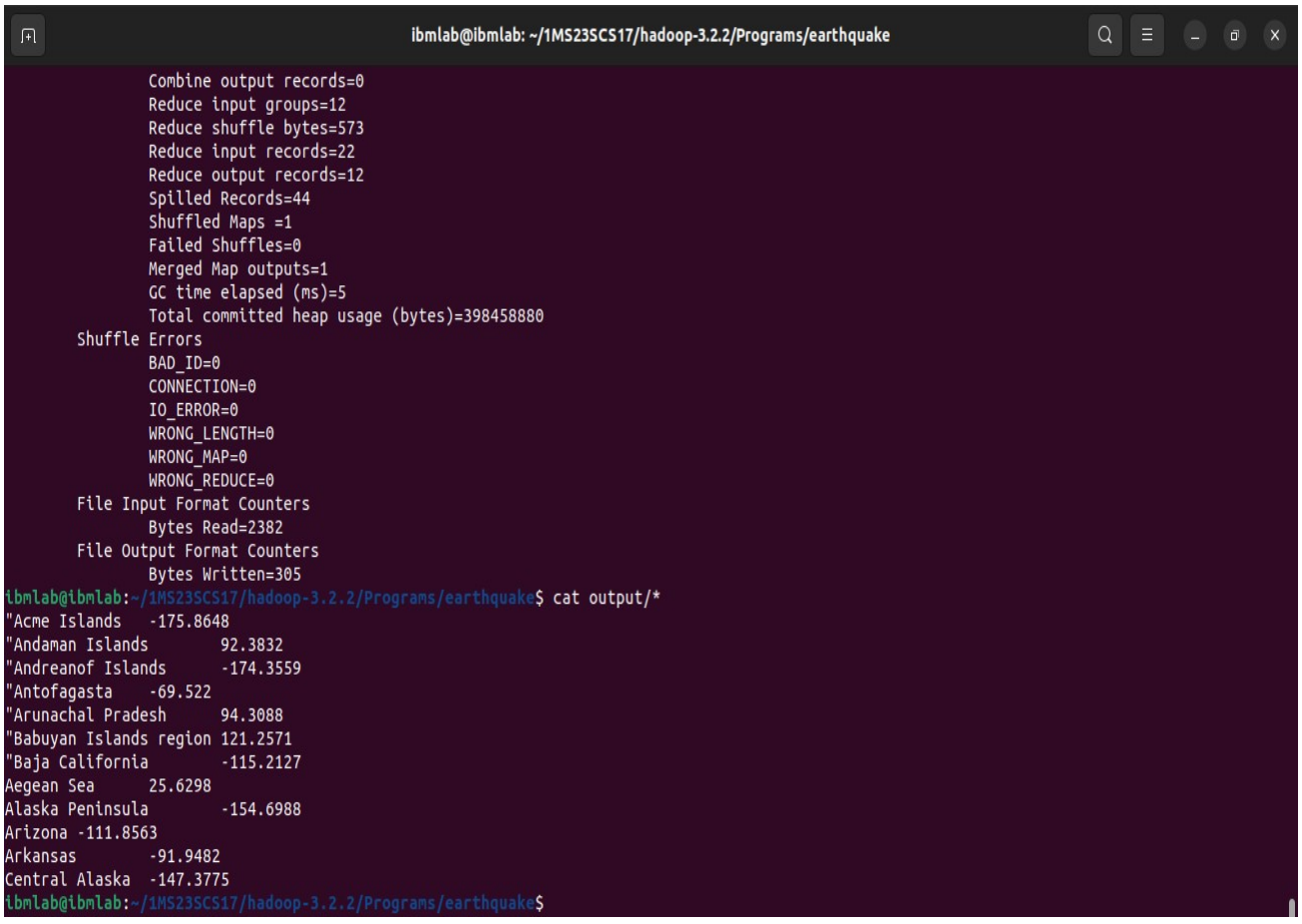
3. To analyse the given **Employee Data** and generate a statistics report with the total number of Female and Male Employees and their average Salary.

```
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=4
                Total committed heap usage (bytes)=398458880
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=8092
        File Output Format Counters
                Bytes Written=94
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/employee$
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/employee$
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/employee$
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/employee$ cat output/*
F Average       7117.073170731707
F Count 41.0
M Average       6333.781194029851
M Count 67.0
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/employee$
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/employee$
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/employee$
```

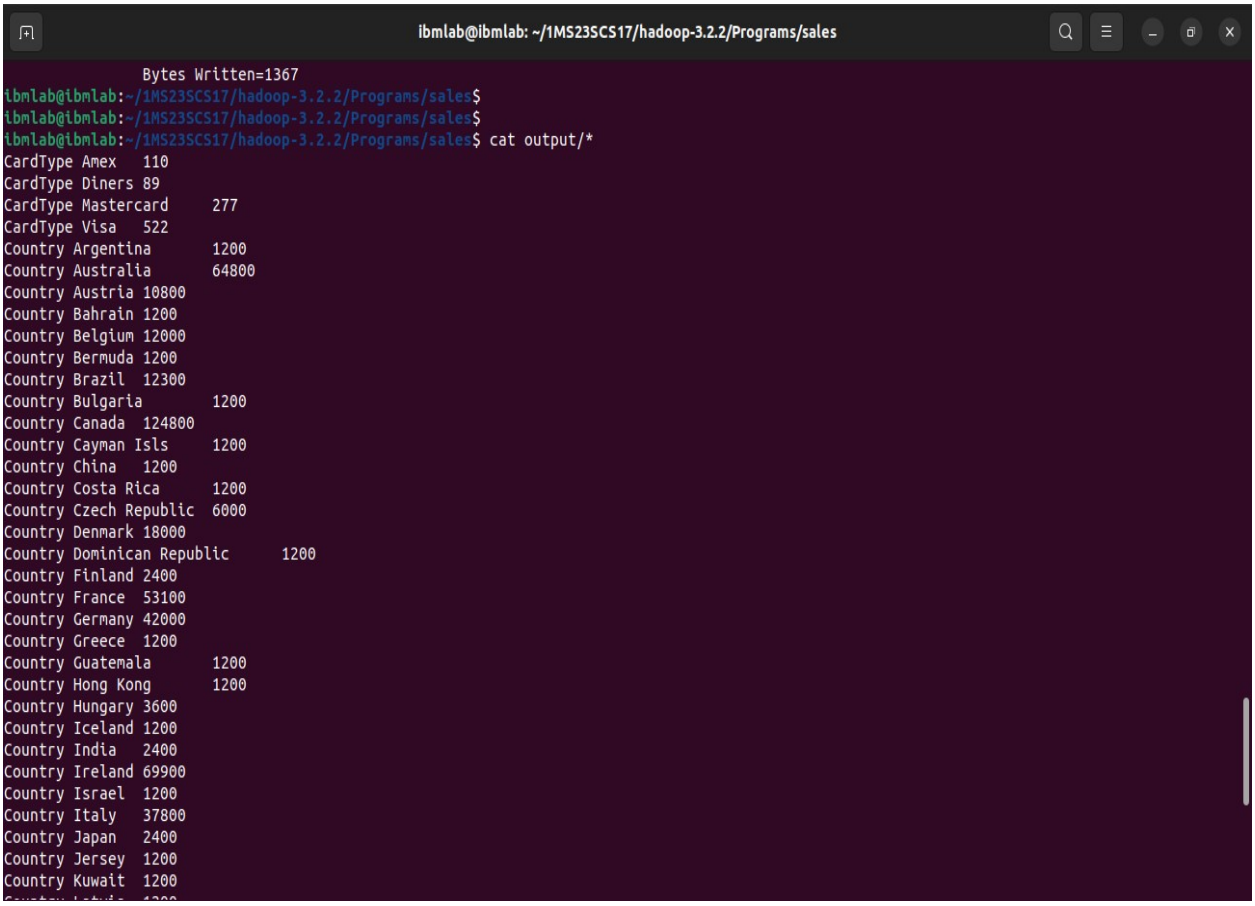
4. To analyse the **Titanic Ship Data** and find the average age of the people (both male and female) who died in the tragedy and also how many people are survived in each class.\

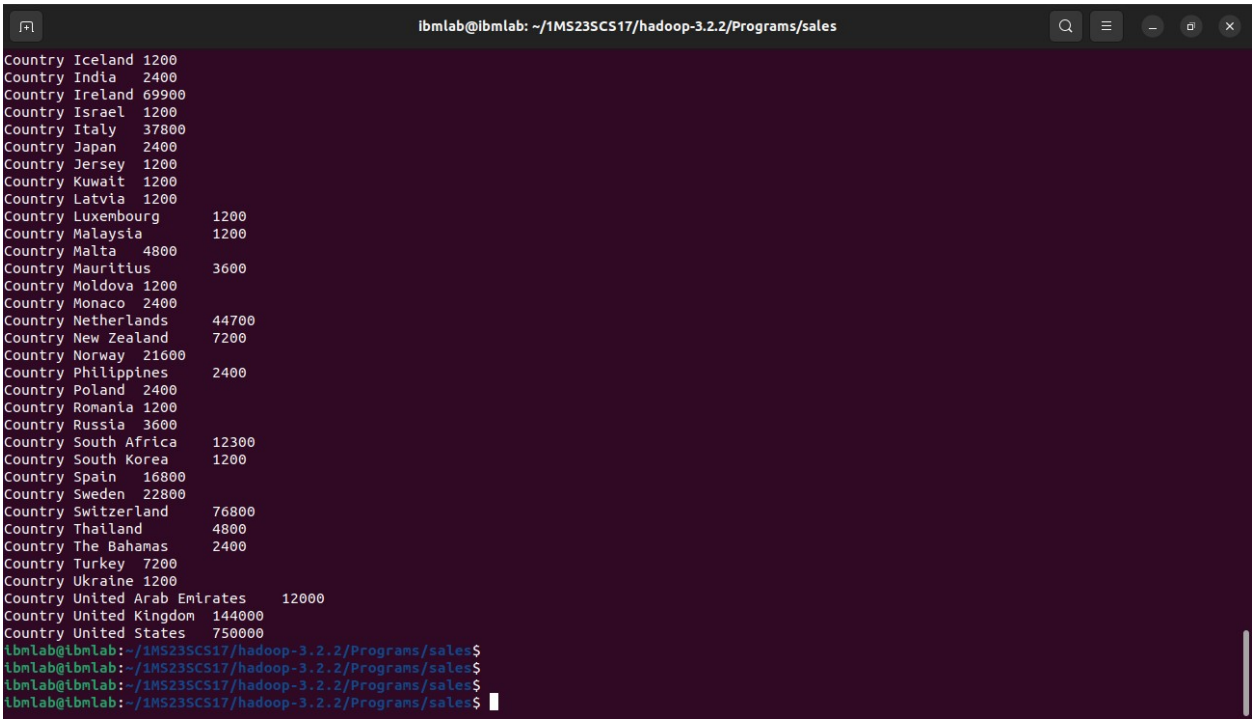


5. To analyse the **Earthquake Data** and generate statistics with region and magnitude/region and depth/region and latitude/region and longitude.



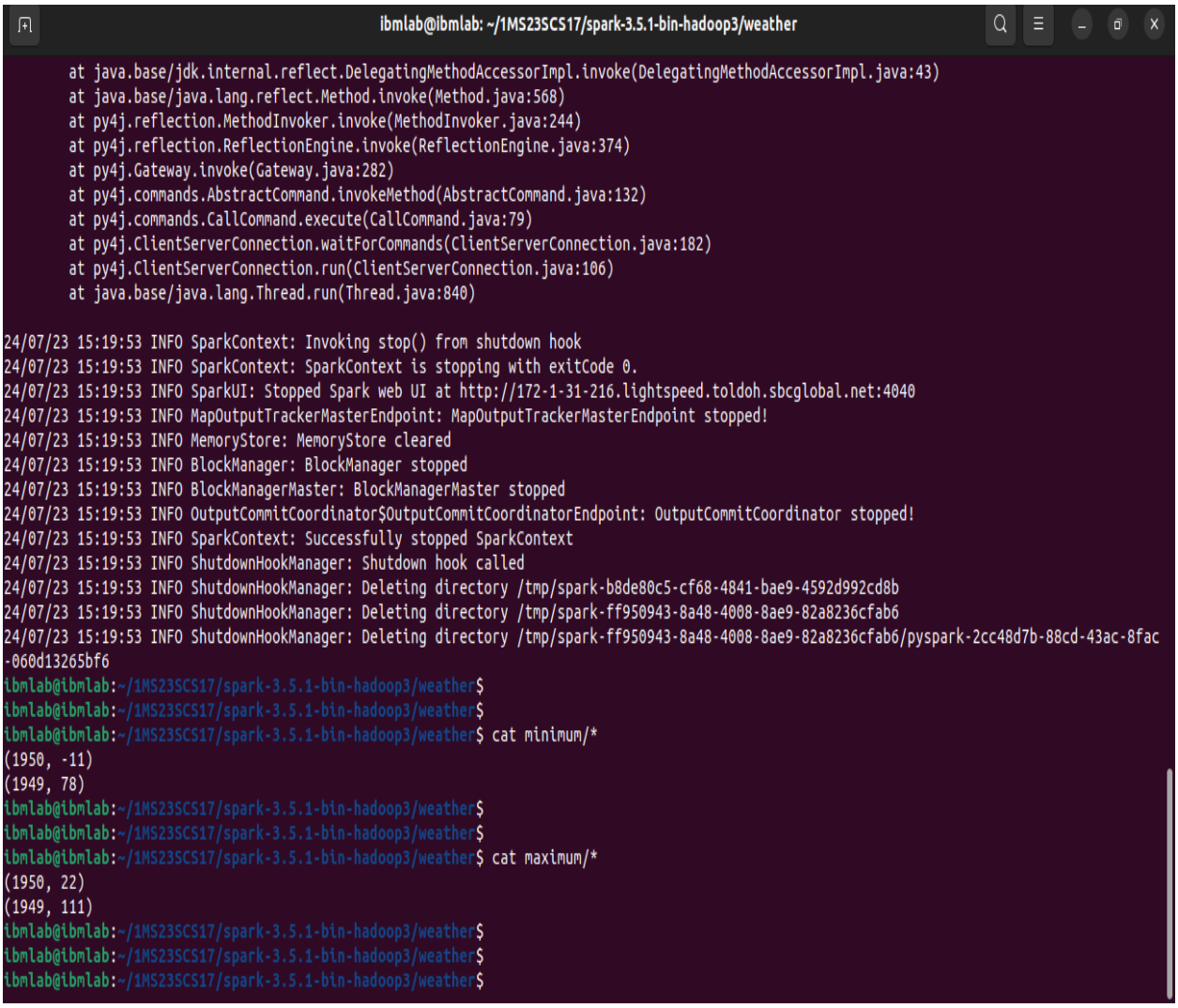
6. To analyse the given **Sales Records** over a period of time and generate data about the country’s total sales and the total number of products. Country’s total sales and the frequency of the payment mode.


ibmlab@ibmlab: ~/1MS23SCS17/hadoop-3.2.2/Programs/sales
Bytes Written=1367
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/sales$
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/sales$
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/sales$ cat output/*
CardType Amex 110
CardType Diners 89
CardType Mastercard 277
CardType Visa 522
Country Argentina 1200
Country Australia 64800
Country Austria 10800
Country Bahrain 1200
Country Belgium 12000
Country Bermuda 1200
Country Brazil 12300
Country Bulgaria 1200
Country Canada 124800
Country Cayman Isls 1200
Country China 1200
Country Costa Rica 1200
Country Czech Republic 6000
Country Denmark 18000
Country Dominican Republic 1200
Country Finland 2400
Country France 53100
Country Germany 42000
Country Greece 1200
Country Guatemala 1200
Country Hong Kong 1200
Country Hungary 3600
Country Iceland 1200
Country India 2400
Country Ireland 69900
Country Israel 1200
Country Italy 37800
Country Japan 2400
Country Jersey 1200
Country Kuwait 1200


ibmlab@ibmlab: ~/1MS23SCS17/hadoop-3.2.2/Programs/sales
Country Iceland 1200
Country India 2400
Country Ireland 69900
Country Israel 1200
Country Italy 37800
Country Japan 2400
Country Jersey 1200
Country Kuwait 1200
Country Latvia 1200
Country Luxembourg 1200
Country Malaysia 1200
Country Malta 4800
Country Mauritius 3600
Country Moldova 1200
Country Monaco 2400
Country Netherlands 44700
Country New Zealand 7200
Country Norway 21600
Country Philippines 2400
Country Poland 2400
Country Romania 1200
Country Russia 3600
Country South Africa 12300
Country South Korea 1200
Country Spain 16800
Country Sweden 22800
Country Switzerland 76800
Country Thailand 4800
Country The Bahamas 2400
Country Turkey 7200
Country Ukraine 1200
Country United Arab Emirates 12000
Country United Kingdom 144000
Country United States 750000
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/sales$
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/sales$
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/sales$
ibmlab@ibmlab:~/1MS23SCS17/hadoop-3.2.2/Programs/sales$

PART-B

7. Write a **Spark program using Python**, to analyse the given **Weather Report Data** and to generate a report with cities having maximum and minimum temperature for a particular year.



8. Write a **Spark program using Python**, to analyse the given **Insurance Data** and generate a statistics report with the construction building name and the count of building/ county name and its frequency.

```
ibmlab@ibmlab:~/1MS23SCS17/spark-3.5.1-bin-hadoop3/insurance$
ibmlab@ibmlab:~/1MS23SCS17/spark-3.5.1-bin-hadoop3/insurance$
ibmlab@ibmlab:~/1MS23SCS17/spark-3.5.1-bin-hadoop3/insurance$
ibmlab@ibmlab:~/1MS23SCS17/spark-3.5.1-bin-hadoop3/insurance$ cat construction/*
('Wood', 17)
('Reinforced Masonry', 2)
('Reinforced Concrete', 3)
('Masonry', 2)
ibmlab@ibmlab:~/1MS23SCS17/spark-3.5.1-bin-hadoop3/insurance$
ibmlab@ibmlab:~/1MS23SCS17/spark-3.5.1-bin-hadoop3/insurance$
ibmlab@ibmlab:~/1MS23SCS17/spark-3.5.1-bin-hadoop3/insurance$ cat county/*
('ALACHUA COUNTY', 24)
ibmlab@ibmlab:~/1MS23SCS17/spark-3.5.1-bin-hadoop3/insurance$
ibmlab@ibmlab:~/1MS23SCS17/spark-3.5.1-bin-hadoop3/insurance$
ibmlab@ibmlab:~/1MS23SCS17/spark-3.5.1-bin-hadoop3/insurance$
```

Pig

1. Write Pig Latin scripts for Crop Production Dataset.

crop_prod = LOAD 'crop_production.csv' USING PigStorage(',') AS (State_Name:chararray, District_Name:chararray, Crop_Year:int, Season:chararray, Crop:chararray, Area:float, Production:float);

DESCRIBE crop_prod;

```
grunt> crop_prod = LOAD 'crop_production.csv' USING PigStorage(',') AS (State_Na
me:chararray, District_Name:chararray, Crop_Year:int, Season:chararray, Crop:cha
rarray, Area:float, Production:float);
grunt> DESCRIBE crop_prod;
crop_prod: {State_Name: chararray,District_Name: chararray,Crop_Year: int,Season
: chararray,Crop: chararray,Area: float,Production: float}
```

a. Calculate total production of each crop.

total_production = GROUP crop_prod BY Crop;

sum_production = FOREACH total_production GENERATE group AS Crop, SUM(crop_prod.Production) AS Total_Production;

DUMP sum_production;

```
(Cowpea(Lobia),240638.0)
(Small millets,5630375.600000024)
(Oilseeds total,4.386756E7)
(Rajmash Kholar,18590.0)
(other oilseeds,4769908.810001001)
(Cond-spcs other,2260.4000107347965)
(Other Dry Fruit,0.0)
(Total foodgrain,4.3270757E7)
(Other Vegetables,963212.0)
(Moong(Green Gram),1.830318779909374E7)
(Peas  (vegetable),0.0)
(Rapeseed &Mustard,9.086926582185636E7)
(Other  Rabi pulses,4805261.400009513)
(Other Citrus Fruit,0.0)
(Other Fresh Fruits,399443.0)
(Ricebean (nagadal),5230.0)
(other misc. pulses,9704.219999995083)
(Arcanut (Processed),192831.0)
(Cashewnut Processed,8121.0)
(Other Kharif pulses,4349860.380000353)
(Peas & beans (Pulses),8752955.239967406)
(Other Cereals & Millets,1186045.6999955177)
(Beans & Mutter(Vegetable),211359.0)
```

b. Find the average production per year for each crop.

```
grouped_by_crop_year = GROUP crop_prod BY (Crop, Crop_Year);

average_production = FOREACH grouped_by_crop_year GENERATE group.Crop AS Crop, group.Crop_Year AS Crop_Year, AVG(crop_prod.Production) AS Avg_Production;

DUMP average_production;
```

```
(Peas & beans (Pulses),2010,1770.018237082067)
(Peas & beans (Pulses),2011,2126.6575757575756)
(Peas & beans (Pulses),2012,2333.253472222222)
(Peas & beans (Pulses),2013,2209.551211006501)
(Peas & beans (Pulses),2014,1828.0802575073528)
(Other Cereals & Millets,1998,2142.553191489362)
(Other Cereals & Millets,1999,471.42857142857144)
(Other Cereals & Millets,2000,1797.4130434782608)
(Other Cereals & Millets,2001,199.23529411764707)
(Other Cereals & Millets,2002,164.88888888888889)
(Other Cereals & Millets,2003,4401.6)
(Other Cereals & Millets,2004,1745.9638554216867)
(Other Cereals & Millets,2005,2311.50000001395)
(Other Cereals & Millets,2006,1694.3488372093022)
(Other Cereals & Millets,2007,2920.691379307159)
(Other Cereals & Millets,2008,2936.808510638298)
(Other Cereals & Millets,2009,1051.128205128205)
(Other Cereals & Millets,2010,2934.9285714285716)
(Other Cereals & Millets,2011,1716.5405405405406)
(Other Cereals & Millets,2012,1106.1052631578948)
(Other Cereals & Millets,2014,70.00526289720284)
(Beans & Mutter(Vegetable),2002,1020.9277108433735)
(Beans & Mutter(Vegetable),2003,1507.404761904762)
```

c. Filter all crops grown in ‘Karnataka’

```
specific_state = FILTER crop_prod BY State_Name == ‘Karnataka’;

unique_crops = GROUP specific_state BY Crop;

DUMP unique_crops;
```

d. Calculate the total area used for each crop in the year 2010.

```
specific_year = FILTER crop_prod BY Crop_Year == 2010;

total_area = GROUP specific_year BY Crop;

sum_area = FOREACH total_area GENERATE group AS Crop, SUM(specific_year.Area)
AS Total_Area;

DUMP sum_area;
```

```
(Wheat,2.9837101E7)
(Banana,319391.1396789551)
(Barley,677806.0)
(Garlic,124884.0)
(Ginger,4554.0)
(Masoor,1618613.0)
(Potato,1339266.0)
(Khesari,512130.0)
(Linseed,257083.0)
(Sesamum,2091602.8100008965)
(Tapioca,96568.09997558594)
(Tobacco,482107.0)
(Arecanut,399306.49981689453)
(Cardamom,19081.0)
(Coconut ,1255481.125)
(Sannhamp,9221.0)
(Soyabean,9558319.0)
(Turmeric,106831.7799949646)
(Arhar/Tur,4289565.0)
(Cashewnut,255319.7510986328)
(Coriander,401461.0)
(Groundnut,5861460.25)
(Guar seed,3126187.0)
(Safflower,242802.0)
(Sugarcane,4795159.140007984)
(Sunflower,889231.9000000954)
(Dry ginger,101069.92992973328)
(Horse-gram,470891.0)
(Niger seed,240962.0)
(Castor seed,889033.0)
(Black pepper,197783.55114746094)
(Cotton(lint),1.0929723E7)
(Dry chillies,430337.0)
(Sweet potato,38264.0)
(Cowpea(Lobia),100769.0)
(Small millets,554661.0)
(Oilseeds total,31854.0)
(other oilseeds,1300390.0)
(Moong(Green Gram),3506872.9999752045)
(Rapeseed &Mustard,5482169.0)
(Other  Rabi pulses,69217.0)
(Other Kharif pulses,278997.0)
(Peas & beans (Pulses),656383.0)
(Other Cereals & Millets,108567.0)
```

2. Write Pig Latin scripts for **Olympic Athletes and Hosts Datasets**.

athletes = LOAD 'olympic_athletes.csv' USING PigStorage(',') AS (athlete_url: chararray, athlete_full_name: chararray, games_participations: int, first_game: chararray, athlete_year_birth: float, athlete_medals: chararray, bio: chararray);

hosts = LOAD 'olympic_hosts.csv' USING PigStorage(',') AS (game_slug: chararray, game_end_date: chararray, game_start_date: chararray, game_location: chararray, game_name: chararray, game_season: chararray, game_year: int);

DESCRIBE athletes;

DESCRIBE hosts;

```
grunt> athletes = LOAD 'olympic_athletes.csv' USING PigStorage(',') AS (athlete_url: chararray, athlete_full_name: chararray, games_participations: in
t, first_game: chararray, athlete_year_birth: float, athlete_medals: chararray, bio: chararray);
grunt> hosts = LOAD 'olympic_hosts.csv' USING PigStorage(',') AS (game_slug: chararray, game_end_date: chararray, game_start_date: chararray, game_loc
ation: chararray, game_name: chararray, game_season: chararray, game_year: int);
grunt> DESCRIBE athletes;
athletes: {athlete_url: chararray,athlete_full_name: chararray,games_participations: int,first_game: chararray,athlete_year_birth: float,athlete_medal
s: chararray,bio: chararray}
grunt> DESCRIBE hosts;
hosts: {game_slug: chararray,game_end_date: chararray,game_start_date: chararray,game_location: chararray,game_name: chararray,game_season: chararray,
game_year: int}
grunt>
```

I. Filter athletes participated in the “Tokyo 2020” games.

tokyo_2020_athletes = FILTER athletes BY first_game == ‘Tokyo 2020’;

DUMP tokyo_2020_athletes;

```
(https://olympics.com/en/athletes/xinxin-wang,Xinxin WANG,1,Tokyo 2020,1998.0,,)
(https://olympics.com/en/athletes/lidianny-echevarria-benitez,Lidianny ECHEVARRIA BENITEZ,1,Tokyo 2020,1996.0,,)
(https://olympics.com/en/athletes/leila-consuelo-martinez-ortega,Leila Consuelo MARTINEZ ORTEGA,1,Tokyo 2020,1994.0,,)
(https://olympics.com/en/athletes/nadezda-makroguzova,Nadezda MAKROGUZOVA,1,Tokyo 2020,1997.0,,)
(https://olympics.com/en/athletes/svetlana-kholomina,Svetlana KHOLOMINA,1,Tokyo 2020,1997.0,,)
(https://olympics.com/en/athletes/tanja-huberli,Tanja HUBERLI,1,Tokyo 2020,1992.0,,)
(https://olympics.com/en/athletes/nina-betschart,Nina BETSCHART,1,Tokyo 2020,1995.0,,)
(https://olympics.com/en/athletes/kelly-claes,Kelly CLAES,1,Tokyo 2020,1995.0,,)
(https://olympics.com/en/athletes/sarah-sponcil,Sarah SPONCIL,1,Tokyo 2020,1996.0,,)
(https://olympics.com/en/athletes/miki-ishii,Miki ISHII,1,Tokyo 2020,1989.0,,)
(https://olympics.com/en/athletes/megumi-murakami,Megumi MURAKAMI,1,Tokyo 2020,1985.0,,)
(https://olympics.com/en/athletes/katja-stam,Katja STAM,1,Tokyo 2020,1998.0,,)
(https://olympics.com/en/athletes/raisa-schoon,Raisa SCHOON,1,Tokyo 2020,,,)
(https://olympics.com/en/athletes/fernanda-pereyra,Fernanda PEREYRA,1,Tokyo 2020,1991.0,,)
(https://olympics.com/en/athletes/julia-sude,Julia SUDE,1,Tokyo 2020,1987.0,,)
(https://olympics.com/en/athletes/viktoria-orsi-toth,Viktoria ORSI TOTH,1,Tokyo 2020,1990.0,,)
(https://olympics.com/en/athletes/gaudencia-makokha,Gaudencia MAKOKHA,1,Tokyo 2020,1992.0,,)
(https://olympics.com/en/athletes/brackcides-khadambi,Brackcides KHADAMBI,1,Tokyo 2020,1984.0,,)
(https://olympics.com/en/athletes/anders-berntsen-mol,Anders Berntsen MOL,1,Tokyo 2020,1997.0,",)
(https://olympics.com/en/athletes/christian-sandlie-sorum,Christian Sandlie SORUM,1,Tokyo 2020,1995.0,",)
(https://olympics.com/en/athletes/oleg-stoyanovskiy,Oleg STOYANOVSKIY,1,Tokyo 2020,1996.0,",)
(https://olympics.com/en/athletes/cherif-younousse,Cherif YOUNOUSSE,1,Tokyo 2020,1995.0,",)
(https://olympics.com/en/athletes/ahmed-tijan,Ahmed TIJAN,1,Tokyo 2020,1995.0,",)
(https://olympics.com/en/athletes/edgars-tocs,Edgars TOCS,1,Tokyo 2020,1988.0,,)
(https://olympics.com/en/athletes/alvaro-morais-filho,Alvaro MORAIS FILHO,1,Tokyo 2020,1990.0,,)
(https://olympics.com/en/athletes/julius-thole,Julius THOLE,1,Tokyo 2020,1997.0,,)
(https://olympics.com/en/athletes/clemens-wickler,Clemens WICKLER,1,Tokyo 2020,1995.0,,)
(https://olympics.com/en/athletes/ilya-leshukov,Ilya LESHUKOV,1,Tokyo 2020,1995.0,,)
(https://olympics.com/en/athletes/pablo-herrera-allepuz,Pablo HERRERA ALLEPUZ,1,Tokyo 2020,,,)
(https://olympics.com/en/athletes/josue-gaston-gaxiola-leyva,Josue Gaston GAXIOLA LEYVA,1,Tokyo 2020,1997.0,,)
(https://olympics.com/en/athletes/jose-luis-rubio-camargo,Jose Luis RUBIO CAMARGO,1,Tokyo 2020,1996.0,,)
(https://olympics.com/en/athletes/michal-bryl,Michal BRYL,1,Tokyo 2020,1994.0,,)
(https://olympics.com/en/athletes/tri-bourne,Tri BOURNE,1,Tokyo 2020,1989.0,,)
(https://olympics.com/en/athletes/adrian-heidrich,Adrian HEIDRICH,1,Tokyo 2020,1994.0,,)
(https://olympics.com/en/athletes/mirco-gerson,Mirco GERSON,1,Tokyo 2020,1992.0,,)
(https://olympics.com/en/athletes/nicolas-capogrosso,Nicolas CAPOGROSSO,1,Tokyo 2020,1995.0,,)
(https://olympics.com/en/athletes/julian-amado-azaad,Julian Amado AZAAD,1,Tokyo 2020,1990.0,,)
(https://olympics.com/en/athletes/christopher-mchugh,Christopher MCHUGH,1,Tokyo 2020,1989.0,,)
(https://olympics.com/en/athletes/damien-schumann,Damien SCHUMANN,1,Tokyo 2020,1987.0,,)
(https://olympics.com/en/athletes/ondrej-perusic,Ondrej PERUSIC,1,Tokyo 2020,1994.0,,)
(https://olympics.com/en/athletes/david-schweiner,David SCHWEINER,1,Tokyo 2020,1994.0,,)
(https://olympics.com/en/athletes/enrico-rossi,Enrico ROSSI,1,Tokyo 2020,1993.0,,)
(https://olympics.com/en/athletes/mohammed-abicha,Mohammed ABICHA,1,Tokyo 2020,1980.0,,)
(https://olympics.com/en/athletes/zouheir-elgraoui,Zouheir ELGRAOUI,1,Tokyo 2020,1994.0,,)
```

II. Filter the games held in “China”.

```
games_in_china = FILTER hosts BY game_location == 'China';
DUMP games_in_china;
```

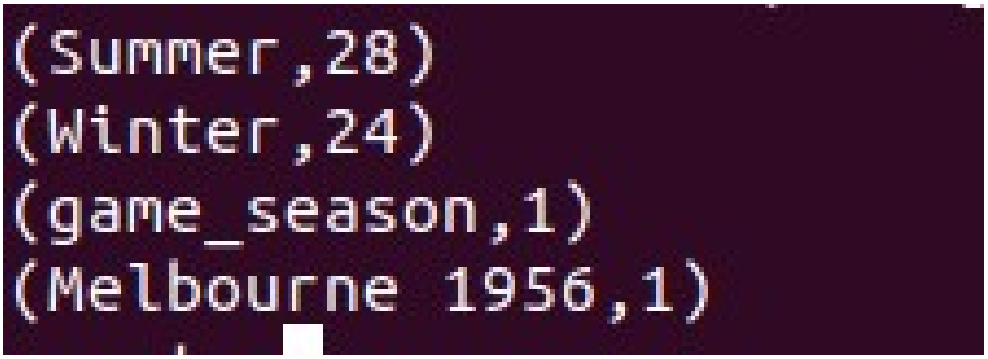
```
(beijing-2022,2022-02-20T12:00:00Z,2022-02-04T15:00:00Z,China,Beijing 2022,Winter,2022)
(beijing-2008,2008-08-24T12:00:00Z,2008-08-08T00:00:00Z,China,Beijing 2008,Summer,2008)
```

III. Group games by season and count the number of games in each session.

```
grouped_by_season = GROUP hosts BY game_season;
```

counted_by_season = FOREACH grouped_by_season GENERATE group AS game_season, COUNT(hosts) AS num_games;

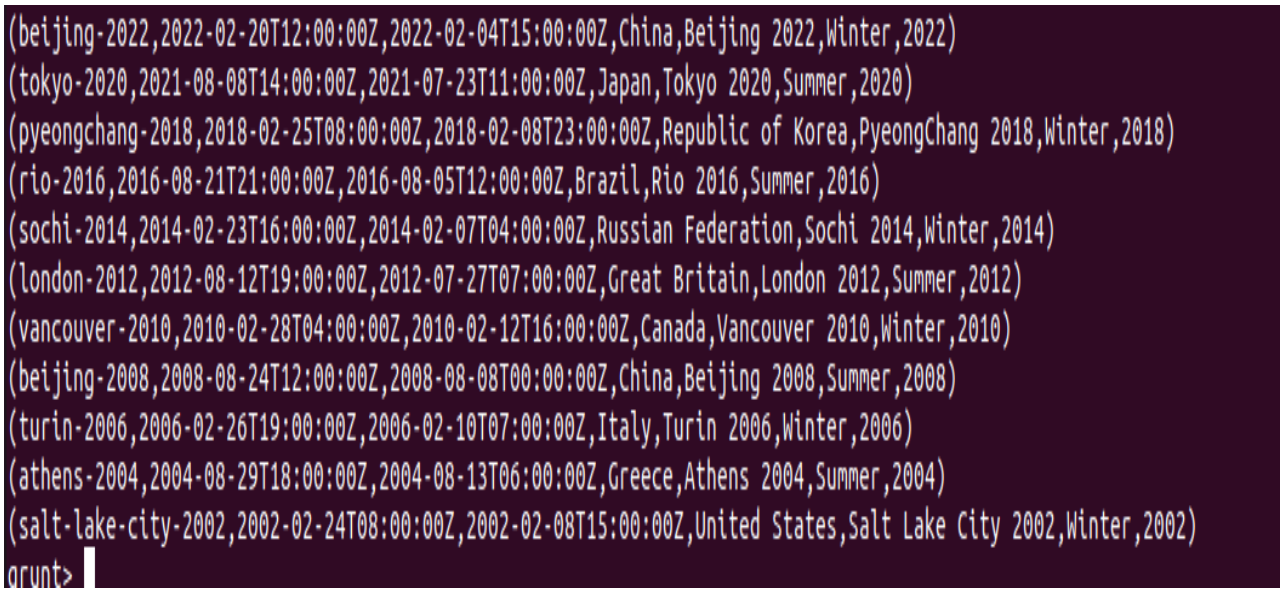
DUMP counted_by_season;



iv. Filter games that occurred after the year 2000.

games_after_2000 = FILTER hosts BY game_year > 2000;

DUMP games_after_2000;



9. Write **Pig Latin scripts** for the following queries on **Olympic Athletes and Hosts Dataset**:
    a. Filter athletes participated in the “Tokyo 2020” games.
    b. Filter the games held in “China”.
    c. Group games by season and count the number of games in each session.
    d. Filter games that occurred after the year 2000.