Name:- Raj Kariya
ID:- 202103048
Google Colab Link:-BigDataLab4.ipynb

Lab-4

Q1) Project and print (empno, name) of employees that are from state 'TX' from employee.csv
Solution:-

Code And Output

Q1). Project and print (empno, name) of employees that are from state 'TX' from employee.csv

```
input = "/content/gdrive/MyDrive/Colab Notebooks/datasets/mr/employee.csv"
lines = sc.textFile(input)
emps = lines.map(lambda line: line.split(","))
emplyee_tx = emps.filter(lambda e: e[4] == "TX")
result = emplyee_tx.map(lambda e: (e[0], e[1]))
print("EmployeeNumber   Name")
for emp in result.collect():
    print(emp)
```

```
EmployeeNumber   Name
('10012', 'Hector Barbossa')
('10200', 'Anton Chigurh')
('10202', 'Jac McKinzie')
```

Q2). Generate List of (empno, name, salary, dep_avg_sal) of employees who have salary > 1.5 times of department average from employee.csv

Solution:-

Code And Output

```
from numpy import average
input = "/content/gdrive/MyDrive/Colab Notebooks/datasets/mr/employee.csv"
lines = sc.textFile(input)
emps = lines.map(lambda line: line.split(","))
```

```
lines = sc.textFile(input)
emps = lines.map(lambda line: line.split(","))
emps_dep = emps.map(lambda e: (e[2], (e[0], e[1], int(e[3]))))
dep_map = emps.map(lambda e: (e[2],(int(e[3]),1)))
dep_map_sumbykey = dep_map.reduceByKey(lambda e1, e2: (e1[0]+e2[0],e1[1]+e2[1]))
average_salary = dep_map_sumbykey.map(lambda e: (e[0], (e[1][0]/e[1][1])))
emps_with_avg_salary = emps_dep.join(average_salary)
# print(emps_with_avg_salary.collect())
# for e in emps_with_avg_salary.collect():
#   print(e[0], e[1][0][2], e[1][1])
emps_with_salary_more_than_avg = emps_with_avg_salary.filter(lambda e: e[1][0][2] > 1.5*e[1][1])
# print(emps_with_salary_more_than_avg.collect())
ans = emps_with_salary_more_than_avg.map(lambda e: (e[1][0][0], e[1][0][1], e[1][0][2] , e[1][1]))
# print(ans)
print("EmployeeNumber   Name    Salary    Dep_Avg_Sal")
for i in ans.collect():
  print(i)
# print(dep_map_sumbykey[1])
#(dep,(sum(salary),count)
# for s in dep_map_sumbykey.collect():
#   print(s[0],s[1])
```

```
EmployeeNumber   Name    Salary    Dep_Avg_Sal
('10019', 'Vito Corleone', 170500, 59967.78846153846)
('10015', 'Jason Foss', 178000, 97064.64)
('10288', 'Peter Monroe', 157000, 97064.64)
('10086', 'Katie Roper', 150290, 97064.64)
('10222', 'Ricardo Ruiz', 148999, 97064.64)
('10010', 'Jennifer Zamora', 220450, 97064.64)
('10272', 'Debra Houlihan', 180000, 68684.0625)
```

Q3) Compute state wise count of employees from employee.csv

Code And Output

Q3) Compute state wise count of employees from employee.csv

```
input = "/content/gdrive/MyDrive/Colab Notebooks/datasets/mr/employee.csv"
lines = sc.textFile(input)
emps = lines.map(lambda line : line.split(","))
emps_state = emps.map(lambda e : (e[4],1))
# print(emps_state.collect())
ans = emps_state.reduceByKey(lambda e1,e2 : e1+e2)
# print(ans.collect())
print("State  Count")
for i in ans.collect():
  print(i)
```

```
State  Count
('MA', 276)
('TX', 3)
('CT', 6)
('VA', 1)
('VT', 2)
('AL', 1)
('WA', 1)
('CA', 1)
('OH', 1)
('IN', 1)
('TN', 1)
('NH', 1)
('RI', 1)
('PA', 1)
('CO', 1)
('NY', 1)
('UT', 1)
('GA', 1)
('FL', 1)
('NC', 1)
('KY', 1)
('ID', 1)
('NV', 1)
('MT', 1)
('OR', 1)
('ND', 1)
('AZ', 1)
('ME', 1)
```

Q4) Compute the Standard Deviation of salary

Solution:-

Code And Output

Q4) Compute the Standard Deviation of salary

```
from math import sqrt
input = "/content/gdrive/MyDrive/Colab Notebooks/datasets/mr/employee.csv"
lines = sc.textFile(input)
emps = lines.map(lambda line : line.split(","))
salary = emps.map(lambda e : int(e[3]))
emps_mean = salary.mean()
emps_count = salary.count()
emps_dev = salary.map(lambda e : pow((e-emps_mean),2)).mean()
print("Standard Deviation of Salary is ",sqrt(emps_dev))
```

```
Standard Deviation of Salary is  25116.159611505238
```

Q5)Compute Department wise Standard Deviation of Salary

Solution:-

Code And Output

```
from math import sqrt
input = "/content/gdrive/MyDrive/Colab Notebooks/datasets/mr/employee.csv"
lines = sc.textFile(input)
emps = lines.map(lambda line : line.split(","))
salary = emps.map(lambda e : (e[2],(int(e[3]),1)))
# print(salary.collect())
depwise = salary.reduceByKey(lambda e1,e2 : (e1[0]+e2[0],e1[1]+e2[1]))
depwise_mean = depwise.map(lambda e: (e[0],e[1][0]/e[1][1]))
Sal_AvgSal = salary.join(depwise_mean)
Diff = Sal_AvgSal.map(lambda e: (e[0],(pow((e[1][0][0] - e[1][1]),2),e[1][0][1])))
# print(Diff.collect())
#Sum(x-xi)^2,count
Sum_count = Diff.reduceByKey(lambda e1,e2: (e1[0]+e2[0],e1[1]+e2[1]))
Dev = Sum_count.map(lambda e: (e[0],sqrt(e[1][0]/e[1][1])))
print("DepartmentNo StdDev_Sal")
for i in Dev.collect():
  print(i)
# depwisemean = depwise.
# emps_mean = salary.mean()
# emps_count = salary.count()
# emps_dev = salary.map(lambda e : pow((e-emps_mean),2)).mean()
# print(sqrt(emps_dev))
```

```
DepartmentNo StdDev_Sal
('4', 8754.93245490792)
('1', 19722.68484258672)
('5', 11420.800779947584)
('3', 32875.83877242373)
('6', 20702.734864531638)
('2', 0.0)
```

Q6) List (empno, dno, name, salary, dept_sal_sd) for employee that are having salary > 1.5 times of SD.

Q6) List (empno, dno, name, salary, dept_sal_sd) for employee that are having salary > 1.5 times of SD.

```
from math import sqrt
input = "/content/gdrive/MyDrive/Colab Notebooks/datasets/mr/employee.csv"
lines = sc.textFile(input)
emps = lines.map(lambda line : line.split(","))
employees = emps.map(lambda e: (e[2],(e[0],e[1],e[3])))
salary = emps.map(lambda e : (e[2],(int(e[3]),1)))
# print(salary.collect())
depwise = salary.reduceByKey(lambda e1,e2 : (e1[0]+e2[0],e1[1]+e2[1]))
depwise_mean = depwise.map(lambda e: (e[0],e[1][0]/e[1][1]))
Sal_AvgSal = salary.join(depwise_mean)
Diff = Sal_AvgSal.map(lambda e: (e[0],(pow((e[1][0][0] - e[1][1]),2),e[1][0][1])))
# print(Diff.collect())
#Sum(x-xi)^2,count
Sum_count = Diff.reduceByKey(lambda e1,e2: (e1[0]+e2[0],e1[1]+e2[1]))
Dev = Sum_count.map(lambda e: (e[0],sqrt(e[1][0]/e[1][1])))
EmpDev = employees.join(Dev).map(lambda e: (e[1][0][0],e[0],e[1][0][1],int(e[1][0][2]),e[1][1]))
# EmpDevCompare = EmpDev.filter(lambda x: x[3]>1.5*x[4])
Emp_Sal_GreaterThanSal = EmpDev.filter(lambda x: (x[3] > 1.5*x[4]))
# print(EmpDev.count())
print("EmployeeNumber Dno  Name        Sal    Dep_Sal_Sd")
for i in Emp_Sal_GreaterThanSal.collect():
  print(i)
```

```
EmployeeNumber Dno  Name        Sal    Dep_Sal_Sd
('10194', '4', 'Colby Andreola', 95660, 8754.93245490792)
('10150', '4', 'Max Cady', 77692, 8754.93245490792)
('10085', '4', 'Judith Carabbio', 93396, 8754.93245490792)
('10155', '4', 'Keyla Del Bosque', 101199, 8754.93245490792)
('10290', '4', 'Susan Exantus', 99280, 8754.93245490792)
('10110', '4', 'Sandra Martin', 105688, 8754.93245490792)
('10005', '4', 'Lucas Patronick', 108987, 8754.93245490792)
('10126', '4', 'Adell Saada', 86214, 8754.93245490792)
('10024', '4', 'Andrew Szabo', 92989, 8754.93245490792)
('10102', '4', 'Edward TRUE', 100416, 8754.93245490792)
('10089', '2', 'Janet King', 250000, 0.0)
('10084', '3', 'Karthikeyan Ait Sidi', 104437, 32875.83877242373)
('10250', '3', 'Alejandro Bacong', 50178, 32875.83877242373)
('10012', '3', 'Hector Barbossa', 92328, 32875.83877242373)
('10245', '3', 'Renee Becker', 110000, 32875.83877242373)
('10199', '3', 'Frank Booth', 103613, 32875.83877242373)
('10082', '3', 'Claudia N Carr', 100031, 32875.83877242373)
('10108', '3', 'Brian Champaigne', 110929, 32875.83877242373)
('10220', '3', 'Rick Clayton', 68678, 32875.83877242373)
('10193', '3', 'Frank Costello', 83552, 32875.83877242373)
('10083', '3', 'Noah Cross', 92329, 32875.83877242373)
('10212', '3', 'Ann Daniele', 85028, 32875.83877242373)
('10197', '3', 'Tommy DeVito', 96820, 32875.83877242373)
('10133', '3', 'Linda Dolan', 70621, 32875.83877242373)
('10028', '3', 'Eric Dougall', 138888, 32875.83877242373)
('10309', '3', 'Boba Fett', 53366, 32875.83877242373)
('10015', '3', 'Jason Foss', 178000, 32875.83877242373)
('10273', '3', 'Lisa Galia', 65707, 32875.83877242373)
```

Q7). Compute how much offset each department's average salary from the overall average

Solution:-

Code And Solution

Q7) Compute how much offset each department's average salary from the overall average.

```
from math import sqrt
input = "/content/gdrive/MyDrive/Colab Notebooks/datasets/mr/employee.csv"
lines = sc.textFile(input)
```

```
lines = sc.textFile(input)
emps = lines.map(lambda line : line.split(","))
employees = emps.map(lambda e: (e[2],(e[0],e[1],e[3])))
salary = emps.map(lambda e : (e[2],(int(e[3]),1)))
Overall_mean = emps.map(lambda e : int(e[3])).mean()
# print(Overall_mean)
# emps_mean = salary.reduceByKey(lambda)
# print(salary.collect())
depwise = salary.reduceByKey(lambda e1,e2 : (e1[0]+e2[0],e1[1]+e2[1]))
depwise_avgSal = depwise.map(lambda e: (e[0],(e[1][0]/e[1][1])))
Offset = depwise_avgSal.map(lambda e:(e[0],(Overall_mean - e[1])))
print("DepID   OffsetSal")
for i in Offset.collect():
  print(i)
# print(Offset.collect())
# depwise_off = depwise.map(lambda e: (e[0],Overall_mean - (e[1][0]/e[1][1])))
# print(depwise_off.collect())
# Sal_AvgSal = salary.join(depwise_mean)
# depwisemean = depwise.
# emps_count = salary.count()
# emps_dev = salary.map(lambda e : pow((e-emps_mean),2)).mean()
# print(sqrt(emps_dev))
```

```
DepID   OffsetSal
('5', 9052.896425921419)
('3', -28043.95511254012)
('4', -27131.415112540126)
('1', -3928.31511254012)
('6', 336.62238745988)
('2', -180979.31511254012)
```

Q8) Computes monthly summary on web access log of Lab01, and compute: (a) Total number of requests. (b) Total download size (in Mega Bytes). It should output: for every month like Dec-2016, Jan-2017, and so!
Solution:-

Code And Output

```
[11] import re
     def month_retrieve(record):
       timestamp = re.split('/',record[3])
       month = re.split(':',timestamp[2])
       year_month = timestamp[1] + " " + month[0]
       download_size = 0
       if(record[9] != '-'):
         download_size = int(record[9])
       return (year_month,(1,download_size))

     input = "/content/gdrive/MyDrive/Colab Notebooks/datasets/mr/web_access_log.txt"
     lines = sc.textFile(input)
     row = lines.map(lambda data: data.split(' '))
     # stores (year_month,(1,down_sz))
     req_kvpairs = row.map(lambda records : month_retrieve(records))
     # print(req_kvpairs.collect())
     # use reduce by key to calculate tot_down size and total records
     TotRequest_Month = req_kvpairs.reduceByKey(lambda e1,e2: (e1[0]+e2[0],e1[1]+e2[1]))
     #give output as month,num_request,download_sz(bytes) , to calculate in bytes we need to divide it by 1024*1024
     Monthly_Summary = TotRequest_Month.map(lambda e:(e[0],(e[1][0],(e[1][1]/(1024)**2))))
     print(f"Month_Year NumberofRequests Download size")
     for i in Monthly_Summary.collect():
       print(i[0], i[1][0], i[1][1])
     # webacsslogs = lines.map(lambda line : re.findall(line))
     # print(webacsslogs.collect())
```

```
Month_Year NumberofRequests Download size
Dec 2015 14148 273.58699226379395
Jan 2016 28224 1755.5749597549438
Feb 2016 64262 1347.5344944000244
```

Q9) List Timestamp , URL of requests in web access for which http response status is 404 .

Solution:-

Code And Output

```python
import re
input = "/content/gdrive/MyDrive/Colab Notebooks/datasets/mr/web_access_log.txt"
# def helper(record):
#   timestamp = re.split('/',record[3])
#   month = re.split(':',timestamp[2])
#   year_month = timestamp[1] + " " + month[0]
#   if(record[5] == 404):
#       url = re.split("",record[7])
#       return (year_month,(1,url))
lines = sc.textFile(input)
row = lines.map(lambda data: data.split(' '))
# print(row.count())
records = row.filter(lambda e: str(e[8]) == "404")
# print(records.collect())
listOfTimeStampUrl = records.map(lambda e:(e[3][1:12],e[10]))
# print(listOfTimeStampUrl.collect())
listOfTimeStampUrl = listOfTimeStampUrl.zipWithIndex().map(lambda e:(e[1]+ 1,e[0]))
listOfTimeStampUrl = listOfTimeStampUrl.filter(lambda e:str(e[1][1]) != "-")
print("Idx   TimeStamp     URL")
for i in listOfTimeStampUrl.collect():
  print(i)
```

```
Idx   TimeStamp     URL
(1, ('12/Dec/2015', '"http://almhuette-raith.at/"'))
(2, ('12/Dec/2015', '"http://www.almhuette-raith.at/"'))
(3, ('12/Dec/2015', '"-"'))
(4, ('13/Dec/2015', '"-"'))
(5, ('13/Dec/2015', '"http://www.almhuette-raith.at/"'))
(6, ('13/Dec/2015', '"-"'))
(7, ('13/Dec/2015', '"http://www.almhuette-raith.at/"'))
(8, ('13/Dec/2015', '"-"'))
(9, ('13/Dec/2015', '"http://www.almhuette-raith.at/"'))
(10, ('13/Dec/2015', '"-"'))
(11, ('13/Dec/2015', '"http://almhuette-raith.at/"'))
(12, ('13/Dec/2015', '"http://almhuette-raith.at/"'))
(13, ('13/Dec/2015', '"http://almhuette-raith.at/"'))
(14, ('13/Dec/2015', '"http://www.almhuette-raith.at/"'))
(15, ('13/Dec/2015', '"http://www.almhuette-raith.at/"'))
(16, ('13/Dec/2015', '"-"'))
(17, ('13/Dec/2015', '"-"'))
(18, ('13/Dec/2015', '"http://www.almhuette-raith.at/index.php?option=com_content&view=article&id=49&Itemid=55"'))
(19, ('13/Dec/2015', '"http://www.almhuette-raith.at/index.php?option=com_content&view=article&id=46&Itemid=54"'))
(20, ('13/Dec/2015', '"http://www.almhuette-raith.at/"'))
(21, ('13/Dec/2015', '"http://almhuette-raith.at/"'))
(22, ('14/Dec/2015', '"http://www.almhuette-raith.at/"'))
(23, ('14/Dec/2015', '"http://www.almhuette-raith.at/"'))
(24, ('14/Dec/2015', '"http://www.almhuette-raith.at/"'))
(25, ('14/Dec/2015', '"http://almhuette-raith.at/"'))
(26, ('15/Dec/2015', '"http://www.almhuette-raith.at/"'))
(27, ('15/Dec/2015', '"-"'))
(28, ('15/Dec/2015', '"http://www.almhuette-raith.at/index.php?option=com_content&view=article&id=49&Itemid=55"'))
(29, ('15/Dec/2015', '"-"'))
(30, ('15/Dec/2015', '"http://www.almhuette-raith.at/index.php?option=com_content&view=article&id=50&Itemid=56"'))
(31, ('15/Dec/2015', '"http://www.almhuette-raith.at/index.php?option=com_content&view=article&id=49&Itemid=55"'))
```