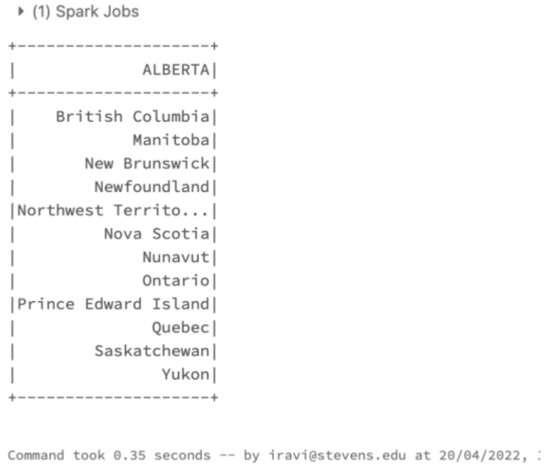
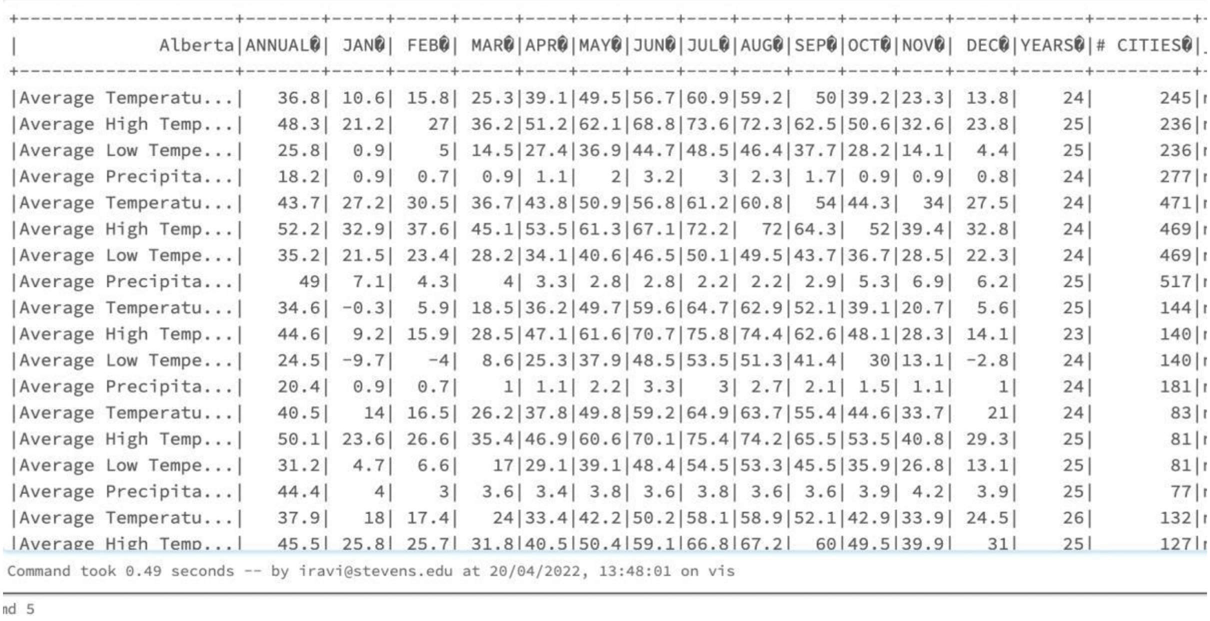
```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, asc, desc
from pyspark.sql.types import IntegerType
from pyspark.sql import functions as F
spark = SparkSession.builder.appName("ishu_assignment").getOrCreate()
sc = spark.sparkContext
sparkDF = spark.read.format("csv").option("header",
"true").load("dbfs:/FileStore/shared_uploads/iravi@stevens.edu/Class_9___12____Data_for_Programming
___Environmental___vshort__1_.csv")

#collecting distinct values of column 0 by excluding empty rows
dist_rows_df = sparkDF.distinct()
dist_val_col0 = dist_rows_df.rdd.map(lambda x : x[0]).collect()

#collecting all the states in states list
states= sparkDF.filter(col("ANNUAL�") == "ANNUAL�").select("ALBERTA")
states.show()

statesList = ['ALBERTA', 'British Columbia', 'Manitoba', 'New Brunswick', 'Newfoundland', 'Northwest
Territories', 'Nova Scotia', 'Nunavut', 'Ontario', 'Prince Edward Island', 'Quebec', 'Saskatchewan', 'Yukon']
```



▸ (1) Spark Jobs

```
+--------------------+
|             ALBERTA|
+--------------------+
|    British Columbia|
|            Manitoba|
|       New Brunswick|
|        Newfoundland|
|Northwest Territo...|
|         Nova Scotia|
|             Nunavut|
|             Ontario|
|Prince Edward Island|
|              Quebec|
|        Saskatchewan|
|               Yukon|
+--------------------+

Command took 0.35 seconds -- by iravi@stevens.edu at 20/04/2022, :
```

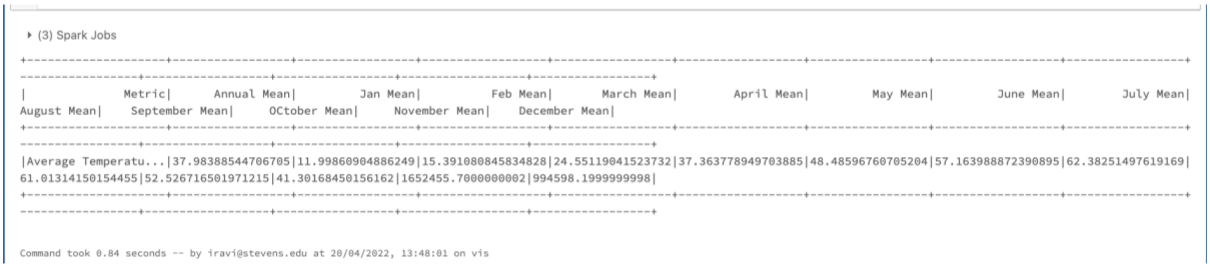
```
sparkDF= sparkDF.filter(col("ANNUAL�") != "ANNUAL�")
sparkDF.show()
```



```
+--------------------+-------+-----+-----+-----+----+----+----+----+----+----+----+----+-----+------+---------+-
|             Alberta|ANNUAL�| JAN�| FEB�| MAR�|APR�|MAY�|JUN�|JUL�|AUG�|SEP�|OCT�|NOV�| DEC�|YEARS�|# CITIES�|
+--------------------+-------+-----+-----+-----+----+----+----+----+----+----+----+----+-----+------+---------+-
|Average Temperatu...|   36.8| 10.6| 15.8| 25.3|39.1|49.5|56.7|60.9|59.2|  50|39.2|23.3| 13.8|    24|      245|
|Average High Temp...|   48.3| 21.2|   27| 36.2|51.2|62.1|68.8|73.6|72.3|62.5|50.6|32.6| 23.8|    25|      236|
|Average Low Tempe...|   25.8|  0.9|    5| 14.5|27.4|36.9|44.7|48.5|46.4|37.7|28.2|14.1|  4.4|    25|      236|
|Average Precipita...|   18.2|  0.9|  0.7|  0.9| 1.1|   2| 3.2|   3| 2.3| 1.7| 0.9| 0.9|  0.8|    24|      277|
|Average Temperatu...|   43.7| 27.2| 30.5| 36.7|43.8|50.9|56.8|61.2|60.8|  54|44.3|  34| 27.5|    24|      471|
|Average High Temp...|   52.2| 32.9| 37.6| 45.1|53.5|61.3|67.1|72.2|  72|64.3|  52|39.4| 32.8|    24|      469|
|Average Low Tempe...|   35.2| 21.5| 23.4| 28.2|34.1|40.6|46.5|50.1|49.5|43.7|36.7|28.5| 22.3|    24|      469|
|Average Precipita...|     49|  7.1|  4.3|    4| 3.3| 2.8| 2.8| 2.2| 2.2| 2.9| 5.3| 6.9|  6.2|    25|      517|
|Average Temperatu...|   34.6| -0.3|  5.9| 18.5|36.2|49.7|59.6|64.7|62.9|52.1|39.1|20.7|  5.6|    25|      144|
|Average High Temp...|   44.6|  9.2| 15.9| 28.5|47.1|61.6|70.7|75.8|74.4|62.6|48.1|28.3| 14.1|    23|      140|
|Average Low Tempe...|   24.5| -9.7|   -4|  8.6|25.3|37.9|48.5|53.5|51.3|41.4|  30|13.1| -2.8|    24|      140|
|Average Precipita...|   20.4|  0.9|  0.7|    1| 1.1| 2.2| 3.3|   3| 2.7| 2.1| 1.5| 1.1|    1|    24|      181|
|Average Temperatu...|   40.5|   14| 16.5| 26.2|37.8|49.8|59.2|64.9|63.7|55.4|44.6|33.7|   21|    24|       83|
|Average High Temp...|   50.1| 23.6| 26.6| 35.4|46.9|60.6|70.1|75.4|74.2|65.5|53.5|40.8| 29.3|    25|       81|
|Average Low Tempe...|   31.2|  4.7|  6.6|   17|29.1|39.1|48.4|54.5|53.3|45.5|35.9|26.8| 13.1|    25|       81|
|Average Precipita...|   44.4|    4|    3|  3.6| 3.4| 3.8| 3.6| 3.8| 3.6| 3.6| 3.9| 4.2|  3.9|    25|       77|
|Average Temperatu...|   37.9|   18| 17.4|   24|33.4|42.2|50.2|58.1|58.9|52.1|42.9|33.9| 24.5|    26|      132|
|Average High Temp...|   45.5| 25.8| 25.7| 31.8|40.5|50.4|59.1|66.8|67.2|  60|49.5|39.9|   31|    25|      127|
Command took 0.49 seconds -- by iravi@stevens.edu at 20/04/2022, 13:48:01 on vis
nd 5
```

```
import numpy as np
temp_prec_col0 =[]#collecting unique values of column 0 in this list and omitting empty rows present in
between the df
for value in dist_val_col0:
    if(not temp_prec_col0.count(value) >0):
        temp_prec_col0.append(value)

#collecting every column values in the following lists respectively (from column 1 to column 15)
annualList = []
JanuaryList = []
febList = []
marchList = []
aprilList = []
mayList = []
juneList = []
julyList = []
augustList = []
septemberList = []
octoberList = []
novemberList = []
decemberList = []
years = []
cities = []
for val in temp_prec_col0:
    if(val == 'Average Temperature (F)'):
        averageTempDF= sparkDF.filter(col("Alberta") == val)
        annualList = averageTempDF.rdd.map(lambda x : x[1]).collect()
        JanuaryList = averageTempDF.rdd.map(lambda x : x[2]).collect()
        febList = averageTempDF.rdd.map(lambda x : x[3]).collect()
        marchList = averageTempDF.rdd.map(lambda x : x[4]).collect()
        aprilList = averageTempDF.rdd.map(lambda x : x[5]).collect()
        mayList = averageTempDF.rdd.map(lambda x : x[6]).collect()
        juneList = averageTempDF.rdd.map(lambda x : x[7]).collect()
        julyList = averageTempDF.rdd.map(lambda x : x[8]).collect()
        augustList = averageTempDF.rdd.map(lambda x : x[9]).collect()
        septemberList = averageTempDF.rdd.map(lambda x : x[10]).collect()
        octoberList = averageTempDF.rdd.map(lambda x : x[11]).collect()
        novemberList = averageTempDF.rdd.map(lambda x : x[12]).collect()
        decemberList = averageTempDF.rdd.map(lambda x : x[13]).collect()
        years = averageTempDF.rdd.map(lambda x : x[14]).collect()
        cities = averageTempDF.rdd.map(lambda x : x[15]).collect()


#since the mean temperature should be calculated based on the weight of cities multiplying each # cities
col value with the respective temperature values collected of each month in the list.
totalYears = 0
totalCities = 0
denominator = 0
annualVal =0
JanuaryVal =0
febVal = 0
marchVal= 0
aprilVal = 0
mayVal = 0
juneVal = 0
julyVal=0
augustVal =0
```

```
septemberVal =0
octoberVal =0
novemberVal =0
decemberVal =0
for i in range(len(annualList)):
   yearsVal = float(years[i])
   citiesVal = float(cities[i])
   stateVal = statesList[i]
   annualVal += float(annualList[i]) * yearsVal * citiesVal
   JanuaryVal += float(JanuaryList[i]) * yearsVal * citiesVal
   febVal += float(febList[i]) * yearsVal * citiesVal
   marchVal += float(marchList[i]) * yearsVal * citiesVal
   aprilVal += float(aprilList[i]) * yearsVal * citiesVal
   mayVal += float(mayList[i]) * yearsVal * citiesVal
   juneVal += float(juneList[i]) * yearsVal * citiesVal
   julyVal += float(julyList[i]) * yearsVal * citiesVal
   augustVal += float(augustList[i]) * yearsVal * citiesVal
   septemberVal += float(septemberList [i]) * yearsVal * citiesVal
   octoberVal += float(octoberList[i]) * yearsVal * citiesVal
   novemberVal += float(novemberList[i]) * yearsVal * citiesVal
   decemberVal += float(decemberList[i]) * yearsVal * citiesVal
   denominator += (yearsVal * citiesVal)
columns = ['Metric', 'Annual Mean', 'Jan Mean', 'Feb Mean', 'March Mean', 'April Mean' ,'May Mean','June Mean','July Mean',
        'August Mean','September Mean','OCtober Mean','November Mean','December Mean']
vals = [('Average Temperature (F)',annualVal/denominator,JanuaryVal/denominator,febVal/denominator,marchVal/denominator,aprilVal/denominator,mayVal/denominator,juneVal/denominator,julyVal/denominator,augustVal/denominator,septemberVal/denominator,octoberVal/denominator,novemberVal,decemberVal)]
mean_temperature_DF = spark.createDataFrame(vals,columns)
mean_temperature_DF.show()
```

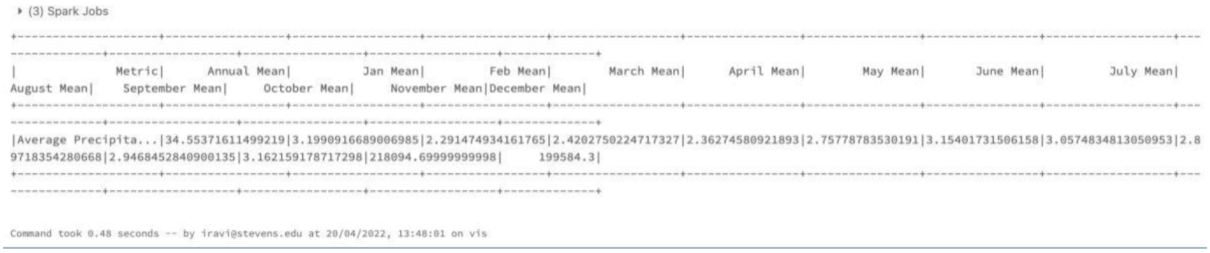


▸ (3) Spark Jobs

```
+--------------------+-----------------+-----------------+------------------+-----------------+------------------+-----------------+------------------+-----------------+
-----------------+------------------+-----------------+------------------+-----------------+
|              Metric|      Annual Mean|         Jan Mean|          Feb Mean|       March Mean|        April Mean|         May Mean|         June Mean|        July Mean|
August Mean|    September Mean|     OCtober Mean|     November Mean|    December Mean|
+--------------------+-----------------+-----------------+------------------+-----------------+------------------+-----------------+------------------+-----------------+
-----------------+------------------+-----------------+------------------+-----------------+
|Average Temperatu...|37.98388544706705|11.99860904886249|15.391080845834828|24.55119041523732|37.363778949703885|48.48596760705204|57.163988872390895|62.38251497619169|
61.01314150154455|52.526716501971215|41.30168450156162|1652455.7000000002|994598.1999999998|
+--------------------+-----------------+-----------------+------------------+-----------------+------------------+-----------------+------------------+-----------------+
-----------------+------------------+-----------------+------------------+-----------------+

Command took 0.84 seconds -- by iravi@stevens.edu at 20/04/2022, 13:48:01 on vis
```

```
#finding mean precipitation of all the months and annual list
annualList = []
JanuaryList = []
febList = []
marchList = []
aprilList = []
mayList = []
juneList = []
julyList = []
augustList = []
septemberList = []
octoberList = []
novemberList = []
decemberList = []
years = []
cities = []
for val in temp_prec_col0:

   if(val == 'Average Precipitation (in)'):
    averagePrecipitationDf= sparkDF.filter(col("Alberta") == val )
    annualList = averagePrecipitationDf.rdd.map(lambda x : x[1]).collect()
    JanuaryList = averagePrecipitationDf.rdd.map(lambda x : x[2]).collect()
    febList = averagePrecipitationDf.rdd.map(lambda x : x[3]).collect()
    marchList = averagePrecipitationDf.rdd.map(lambda x : x[4]).collect()
    aprilList = averagePrecipitationDf.rdd.map(lambda x : x[5]).collect()
    mayList = averagePrecipitationDf.rdd.map(lambda x : x[6]).collect()
    juneList = averagePrecipitationDf.rdd.map(lambda x : x[7]).collect()
    julyList = averagePrecipitationDf.rdd.map(lambda x : x[8]).collect()
    augustList = averagePrecipitationDf.rdd.map(lambda x : x[9]).collect()
    septemberList = averagePrecipitationDf.rdd.map(lambda x : x[10]).collect()
    octoberList = averagePrecipitationDf.rdd.map(lambda x : x[11]).collect()
    novemberList = averagePrecipitationDf.rdd.map(lambda x : x[12]).collect()
    decemberList = averagePrecipitationDf.rdd.map(lambda x : x[13]).collect()
    years = averagePrecipitationDf.rdd.map(lambda x : x[14]).collect()
    cities = averagePrecipitationDf.rdd.map(lambda x : x[15]).collect()

#since the mean precipitation should be calculated based on the weight of cities multiplying each '# cities'
col value with the respective precipitation values collected of each month in the list.
totalYears = 0
totalCities = 0
denominator = 0
annualVal =0
JanuaryVal =0
febVal = 0
marchVal= 0
aprilVal = 0
mayVal = 0
juneVal = 0
julyVal=0
augustVal =0
septemberVal =0
octoberVal =0
novemberVal =0
decemberVal =0
for i in range(len(annualList)):
  yearsVal = float(years[i])
  citiesVal = float(cities[i])
  stateVal = statesList[i]
  annualVal += float(annualList[i]) * yearsVal * citiesVal
  JanuaryVal += float(JanuaryList[i]) * yearsVal * citiesVal
  febVal += float(febList[i]) * yearsVal * citiesVal
  marchVal += float(marchList[i]) * yearsVal * citiesVal
  aprilVal += float(aprilList[i]) * yearsVal * citiesVal
  mayVal += float(mayList[i]) * yearsVal * citiesVal
  juneVal += float(juneList[i]) * yearsVal * citiesVal
  julyVal += float(julyList[i]) * yearsVal * citiesVal
  augustVal += float(augustList[i]) * yearsVal * citiesVal
  septemberVal += float(septemberList [i]) * yearsVal * citiesVal
  octoberVal += float(octoberList[i]) * yearsVal * citiesVal
  novemberVal += float(novemberList[i]) * yearsVal * citiesVal
  decemberVal += float(decemberList[i]) * yearsVal * citiesVal
  denominator += (yearsVal * citiesVal)
columns = ['Metric', 'Annual Mean', 'Jan Mean', 'Feb Mean', 'March Mean', 'April Mean' ,'May Mean','June
Mean','July Mean','August Mean','September Mean','October Mean','November Mean','December Mean']
vals = [('Average Precipitation (in)', annualVal/denominator, JanuaryVal/denominator,
febVal/denominator, marchVal/denominator, aprilVal/denominator,
mayVal/denominator,juneVal/denominator,julyVal/denominator,augustVal/denominator,
septemberVal/denominator,octoberVal/denominator,novemberVal,decemberVal)]
```

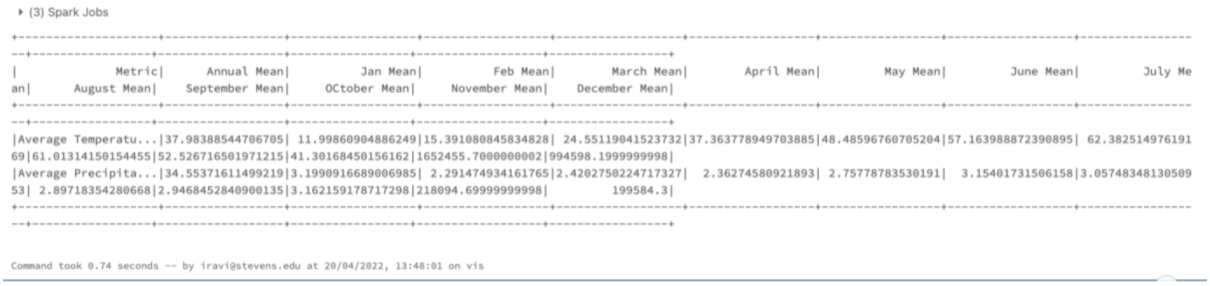
mean_precipitation_DF = spark.createDataFrame(vals,columns)
mean_precipitation_DF.show()



▸ (3) Spark Jobs

```
+--------------------+-----------------+------------------+-----------------+------------------+----------------+----------------+----------------+------------------+---
-------------+------------------+-----------------+------------------+-------------+
|              Metric|      Annual Mean|          Jan Mean|         Feb Mean|        March Mean|      April Mean|        May Mean|       June Mean|         July Mean|
August Mean|   September Mean|      October Mean|    November Mean|December Mean|
+--------------------+-----------------+------------------+-----------------+------------------+----------------+----------------+----------------+------------------+---
-------------+------------------+-----------------+------------------+-------------+
|Average Precipita...|34.55371611499219|3.1990916689006985|2.291474934161765|2.4202750224717327|2.36274580921893|2.75778783530191|3.15401731506158|3.0574834813050953|2.8
9718354280668|2.9468452840900135|3.162159178717298|218094.69999999998|     199584.3|
+--------------------+-----------------+------------------+-----------------+------------------+----------------+----------------+----------------+------------------+---
-------------+------------------+-----------------+------------------+-------------+

Command took 0.48 seconds -- by iravi@stevens.edu at 20/04/2022, 13:48:01 on vis
```

#merging temperature and precipitaion df together
mergedtemp_precep_DF = mean_temperature_DF.union(mean_precipitation_DF)
mergedtemp_precep_DF .show()



▸ (3) Spark Jobs

```
+--------------------+-----------------+------------------+------------------+------------------+------------------+-----------------+------------------+----------------
--+-----------------+------------------+-----------------+------------------+-----------------+
|              Metric|      Annual Mean|          Jan Mean|          Feb Mean|        March Mean|        April Mean|         May Mean|         June Mean|          July Me
an|      August Mean|   September Mean|     OCtober Mean|     November Mean|    December Mean|
+--------------------+-----------------+------------------+------------------+------------------+------------------+-----------------+------------------+----------------
--+-----------------+------------------+-----------------+------------------+-----------------+
|Average Temperatu...|37.98388544706705| 11.99860904886249|15.391080845834828| 24.55119041523732|37.363778949703885|48.48596760705204|57.163988872390895| 62.382514976191
69|61.01314150154455|52.526716501971215|41.30168450156162|1652455.7000000002|994598.1999999998|
|Average Precipita...|34.55371611499219|3.1990916689006985| 2.291474934161765|2.4202750224717327|  2.36274580921893| 2.75778783530191|  3.15401731506158|3.05748348130509
53| 2.89718354280668|2.9468452840900135|3.162159178717298|218094.69999999998|         199584.3|
+--------------------+-----------------+------------------+------------------+------------------+------------------+-----------------+------------------+----------------
--+-----------------+------------------+-----------------+------------------+-----------------+

Command took 0.74 seconds -- by iravi@stevens.edu at 20/04/2022, 13:48:01 on vis
```