**PySpark Capstone**

**1.Load the given data to an RDD and perform data cleaning if required. Remove corrupted (StudentID as ??) and duplicate records.**

We load the data and removed the data and cleaned the duplicate values.

//loaded the data into the path

student=sc.textFile("C:/Users/chandrasehkar.pilla/Downloads/SparkPySpark\_Capstone (1)/SparkPySpark\_Capstone/studentdata.csv")

header=student.first()

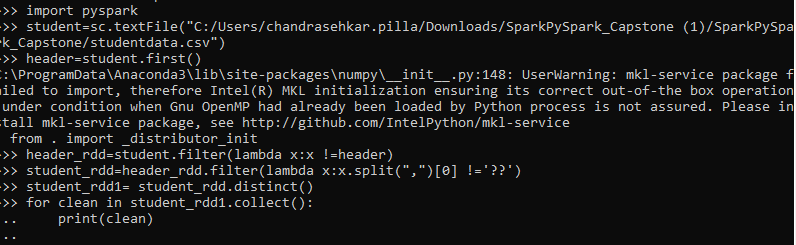
header\_rdd=student.filter(lambda x:x !=header)

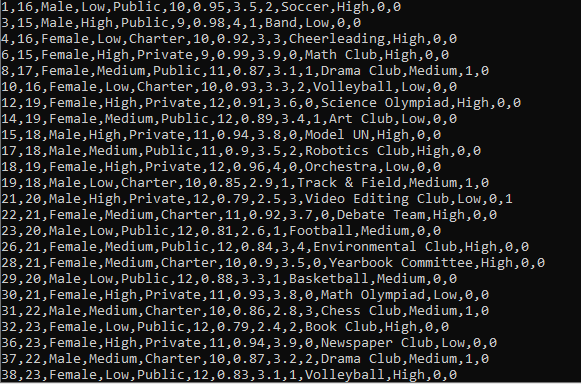
student\_rdd=header\_rdd.filter(lambda x:x.split(",")[0] !='??')

student\_rdd1= student\_rdd.distinct()

for clean in student\_rdd1.collect():

print(clean)



****

**2.Create a paired RDD that stores StudentID and GPA .Sort the output in descending order of GPA.**

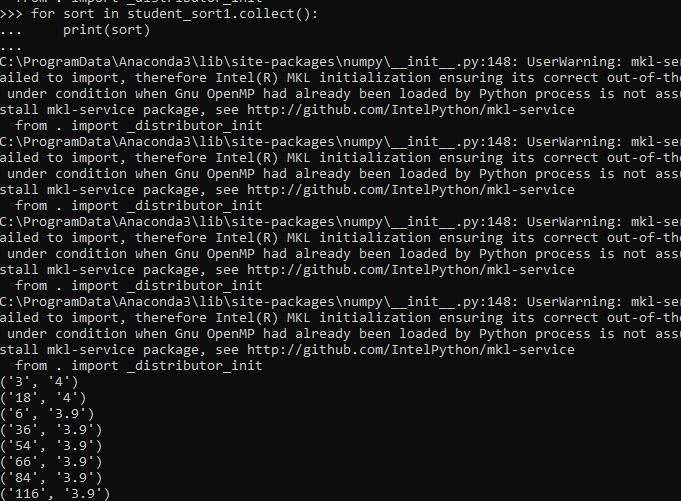
student\_rdd2=student\_rdd1.map(lambda x:(x.split(",")[0],x.split(",")[7]))

student\_sort1=student\_rdd2.sortBy(lambda x:x[1],ascending=False)

for sort in student\_sort1.collect():

print(sort)



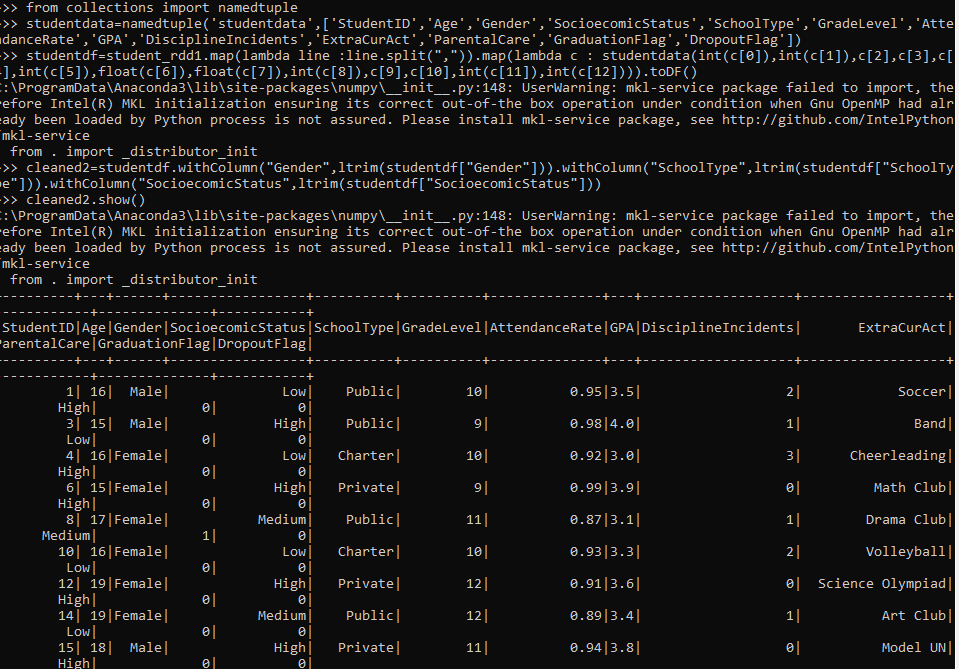


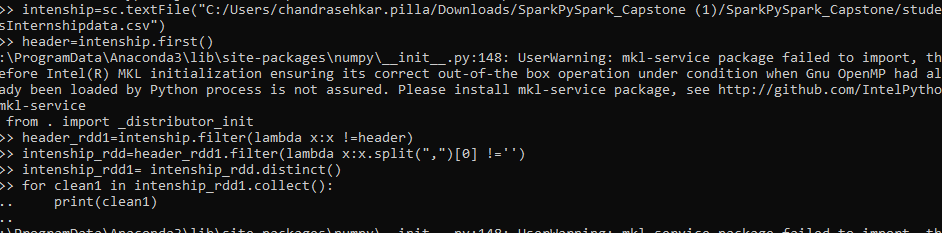
**3.Create a DataFrame in pyspark and use it for futher analysis.**

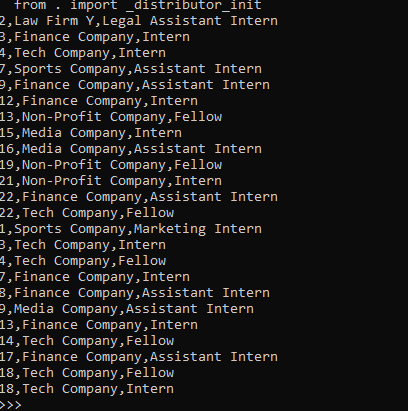
from collections import namedtuple

studentdata=namedtuple('studentdata',['StudentID','Age','Gender','SocioecomicStatus','SchoolType','GradeLevel','AttendanceRate','GPA','DisciplineIncidents','ExtraCurAct','ParentalCare','GraduationFlag','DropoutFlag'])

studentdf=student\_rdd1.map(lambda line :line.split(",")).map(lambda c : studentdata(int(c[0]),int(c[1]),c[2],c[3],c[4],int(c[5]),float(c[6]),float(c[7]),int(c[8]),c[9],c[10],int(c[11]),int(c[12]))).toDF()

****

****

**\**

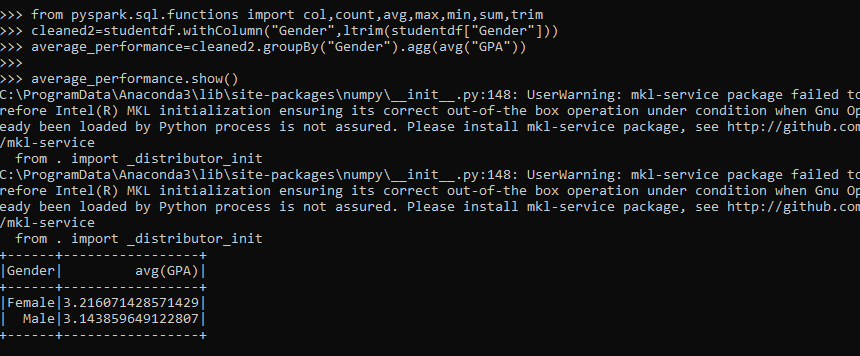
**4.How does average academic performance vary between male and female student?**

rdd2 in the orginal data set the values have some space in the left of the value,so we have to apply the ltrim to it.

from pyspark.sql.functions import col,count,avg,max,min,sum,trim

cleaned2=studentdf.withColumn("Gender",ltrim(studentdf["Gender"]))

average\_performance=studentdf.groupBy("Gender").agg(avg("GPA"))

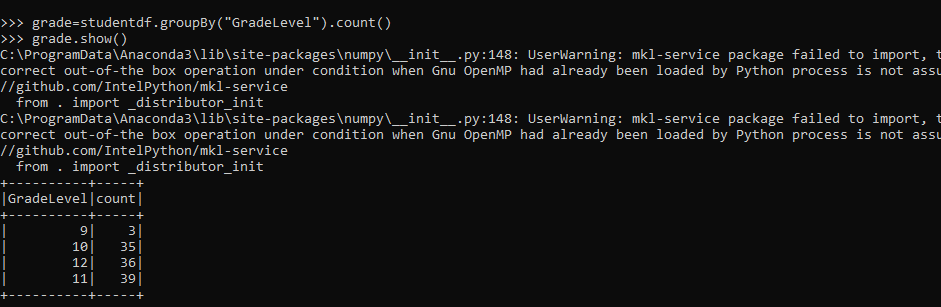
****

**5.what is the student population distributed across different grade levels?**

from pyspark.sql.functions import count

grade=studentdf.groupBy("GradeLevel").count()

grade.show()

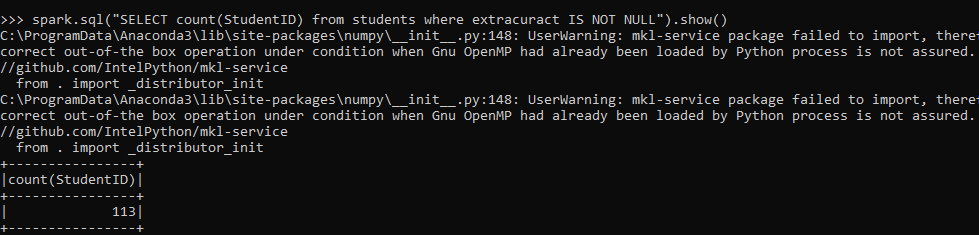
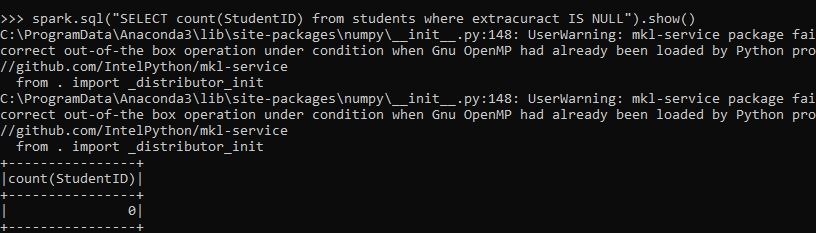
****

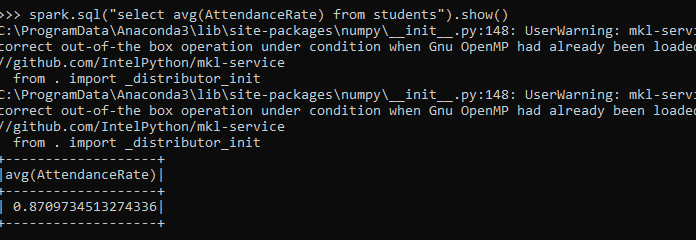
**6.Do students participating in extracurricular activities tend to have better attendance records compared to those not involved [hint:students details with extracurricular activities]**

spark.sql("SELECT count(StudentID) from students where extracuract IS NOT NULL").show()

spark.sql("SELECT count(StudentID) from students where extracuract IS NULL").show()

spark.sql("select avg(AttendanceRate) from students").show()

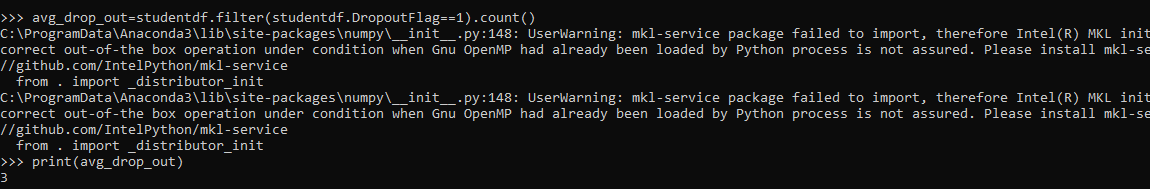
**** ****

****

**\*7.what is the dropout rate among the student population?**

avg\_drop\_out=studentdf.filter(studentdf.DropoutFlag==1).count()

print(avg\_drop\_out)

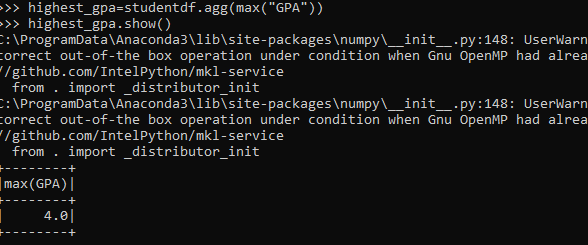
****

**8.what is the highest academic achievement level among the student population(hint:highest GPA)**

from pyspark.sql.functions import max

highest\_gpa=studentdf.agg(max("GPA"))

print(highest\_gpa)

****

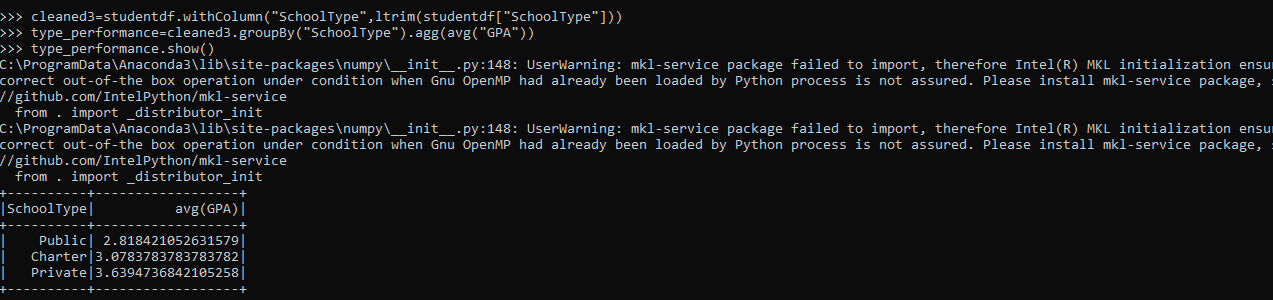
**9.** How does academaic performance vary across different acholl types(pubilc,private,charater)?

from pyspark.sql.functions import avg,ltrim

cleaned3=studentdf.withColumn("SchoolType",ltrim(studentdf["SchoolType"]))

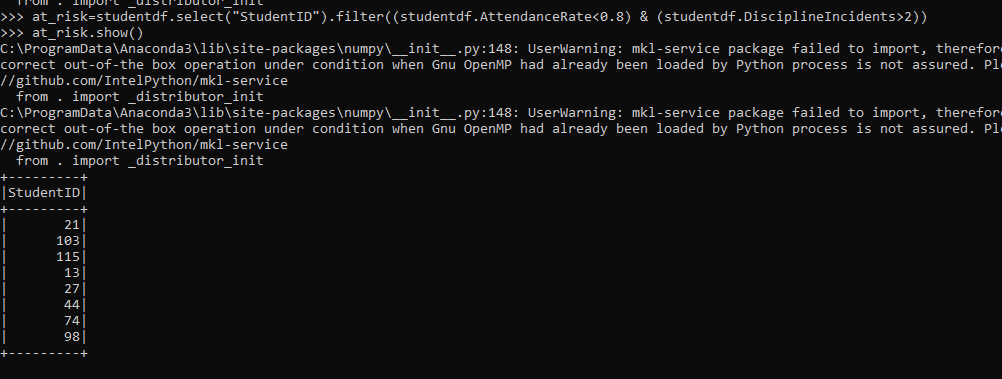
type\_performance=cleaned3.groupBy("SchoolType").agg(avg("GPA"))

type\_performance.show()

****

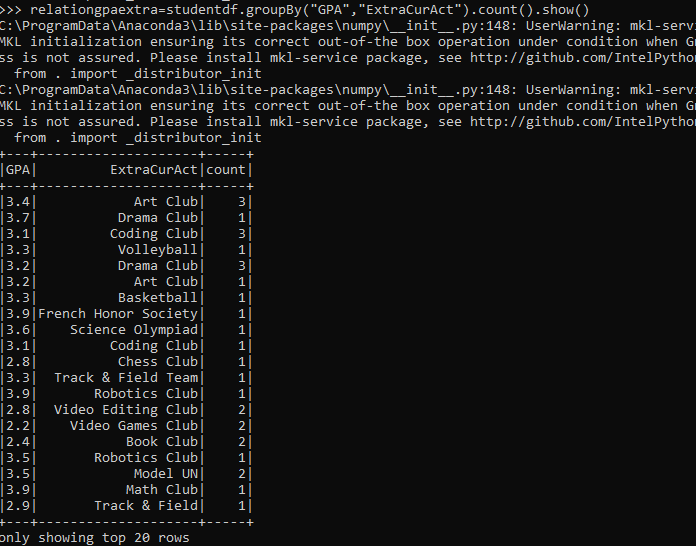
**\*10.Identify students with low attendance rates and multiple discipilanary actions as potential at-risk students?**

at\_risk=studentdf.orderBy( “AttendanceRate”).filter(studentdf.DisciplineIncidents>=2).show()

****

**11.Is there a correlation between partition in extracurricular activities and academaic performance?**

relationgpaextra=studentdf.groupBy("GPA","ExtraCurAct").count().show()

****

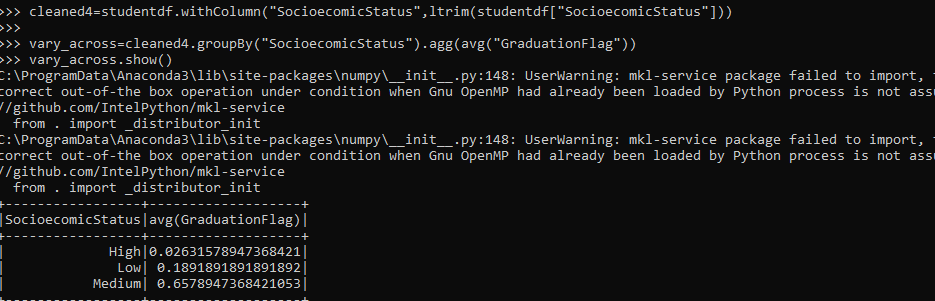
**12.**  **How does the graduaction rate vary across different socioeconomic status levels?**

from pyspark.sql.functions import avg,ltrimn

cleaned4=studentdf.withColumn("SocioecomicStatus",ltrim(studentdf["SocioecomicStatus"]))

vary\_across=cleaned4.groupBy("SocioecomicStatus").agg(avg("GraduationFlag"))

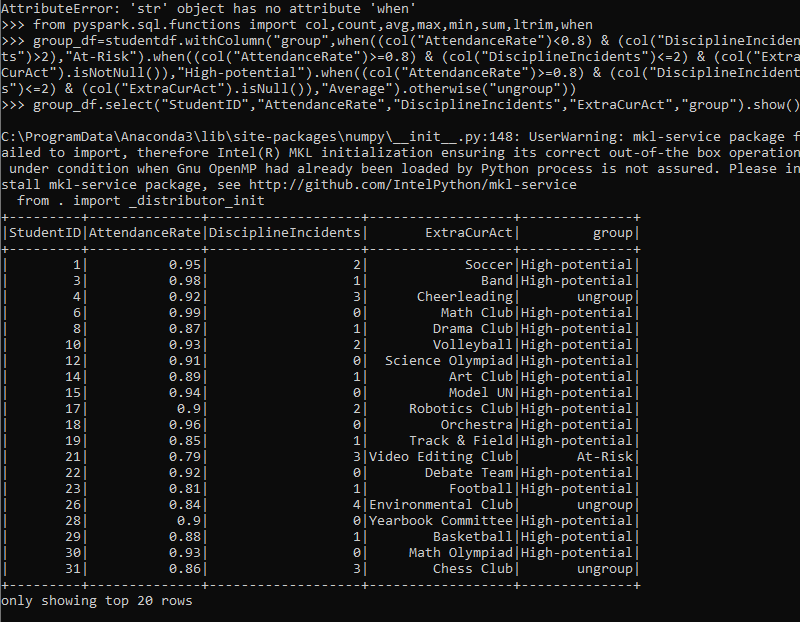
vary\_across.show()

****

**13.Segment students into different groups based on their attendance ,discipline incidents and extracurricular activities.**

group\_df=studentdf.withColumn("group",when((col("AttendanceRate")<0.8) & (col("DisciplineIncidents")>2),"At-Risk").when((col("AttendanceRate")>=0.8) & (col("DisciplineIncidents")<=2) & (col("ExtraCurAct").isNotNull()),"High-potential").when((col("AttendanceRate")>=0.8) & (col("DisciplineIncidents")<=2) & (col("ExtraCurAct").isNull()),"Average").otherwise("ungroup"))

group\_df.select("StudentID","AttendanceRate","DisciplineIncidents","ExtraCurAct","group").show()

****

**14.Consider Internship dataset and count numbers of empty lines using accumulators.Remove the empty lines and save the cleaned data as parquet?**

intenship=sc.textFile("C:/Users/chandrasehkar.pilla/Downloads/SparkPySpark\_Capstone/SparkPySpark\_Capstone/studentsInternshipdata.csv")

Emptylines=sc.accumulator(0)

intenship.foreach(lambda line: Emptylines.add(1) if len(line) == 0 else None)

print(Emptylines.value)

header=intenship.first()

header\_rdd1=intenship.filter(lambda x:x !=header)

intenship\_rdd=header\_rdd1.filter(lambda x:x.split(",")[0] !='')

intenship\_rdd1= intenship\_rdd.distinct()

for clean1 in intenship\_rdd1.collect():

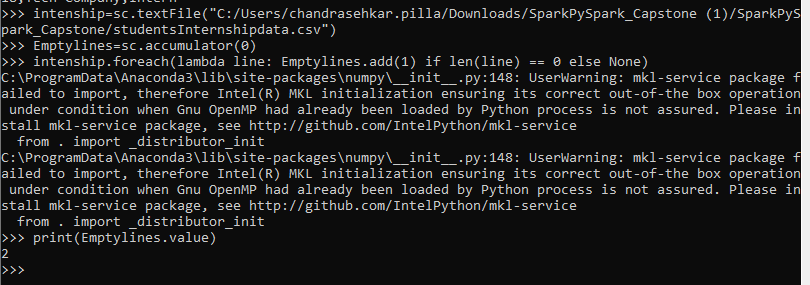
... print(clean1)

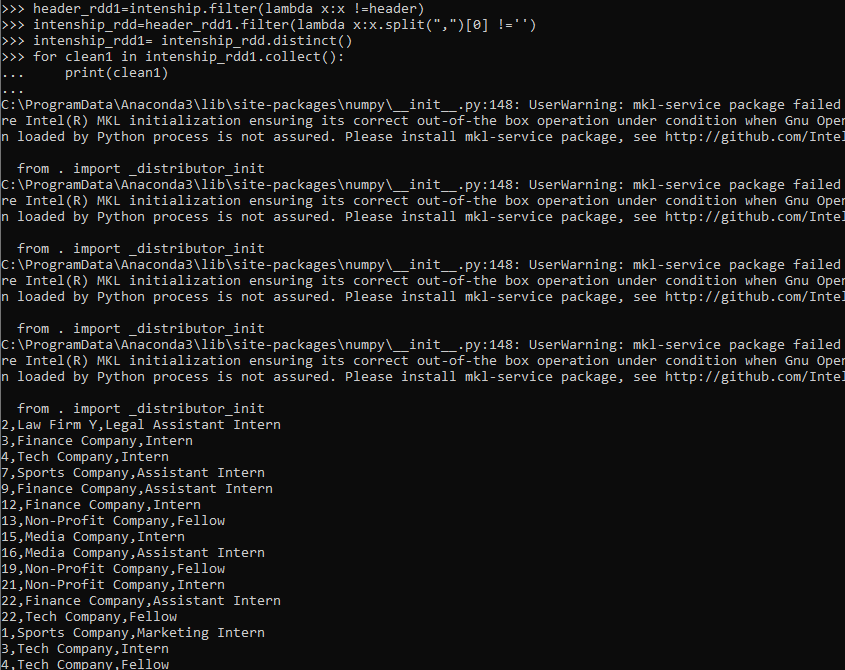
from collections import namedtuple

intenshipdata=namedtuple('intenshipdata',['StudentID','InternshipCompany','Role'])

intenshipdf=intenship\_rdd1.map(lambda x:x.split(",")).map(lambda c :intenshipdata(int(c[0]),c[1],c[2])).toDF()

intenshipdf.write.parquet("C:/Users/chandrasehkar.pilla/Downloads/SparkPySpark\_Capstone/SparkPySpark\_Capstone/output")

****

****

****

**15.Create a DataFrame from cleaned Internship data and find out the students who got multiple internships.**

from collections import namedtuple

intenshipdata=namedtuple('intenshipdata',['StudentID','Internship','Role'])

intenshipdf=intenship\_rdd1.map(lambda x:x.split(",")).map(lambda c :intenshipdata(int(c[0]),c[1],c[2])).toDF()

from pyspark.sql.functions import col,count

multipleintenship=intenshipdf.groupBy("StudentID").agg(count("InternshipCompany").alias("InternshipCount")).filter(col("InternshipCount")>=2).show()

****

**16.Broadcast Internship data and use it as look up while processing large StudentsData,Processed final Dataframe should contain the fields StudentsID,SocioeconomicStatus,GradeLevel,InternshipCompany,Role**

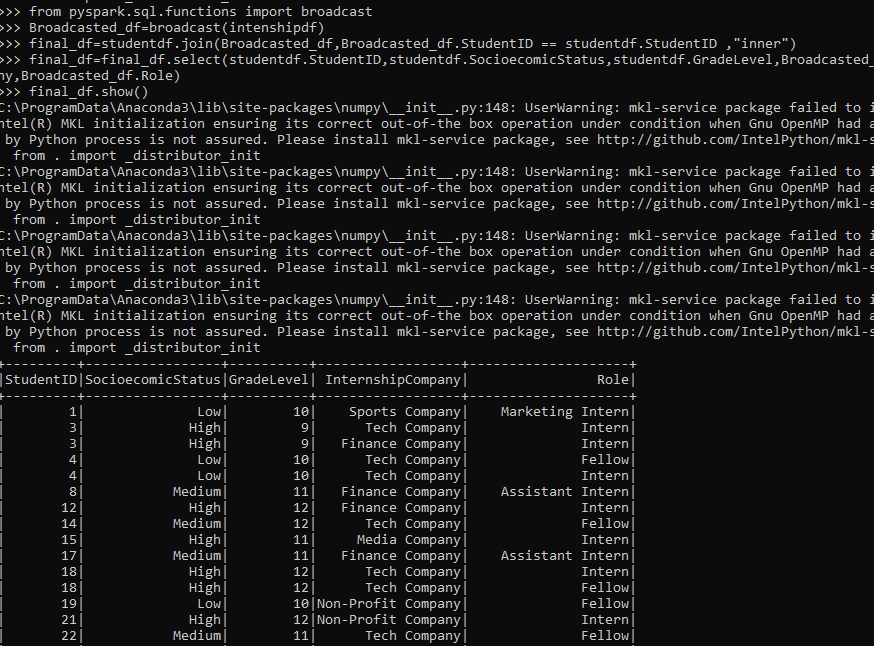
from pyspark.sql.functions import broadcast

Broadcasted\_df=broadcast(intenshipdf)

final\_df=studentdf.join(Broadcasted\_df,Broadcasted\_df.StudentID == studentdf.StudentID ,"inner")

final\_df=final\_df.select(studentdf.StudentID,studentdf.SocioecomicStatus,studentdf.GradeLevel,Broadcasted\_df.InternshipCompany,Broadcasted\_df.Role)

final\_df.show()

****

**17.create two dataFrames and Implement join operation between Students data and Internship data and calculate SocioeconomicStatus internships achievied?**

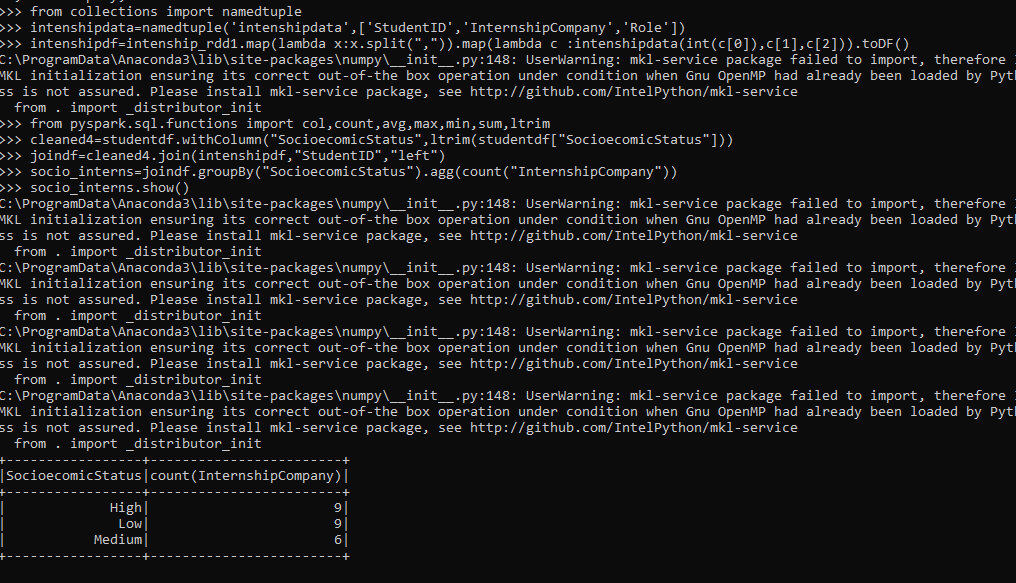
from pyspark.sql.functions import col,count,avg,max,min,sum,ltrim

cleaned4=studentdf.withColumn("SocioecomicStatus",ltrim(studentdf["SocioecomicStatus"]))

joindf=cleaned4.join(intenshipdf,"StudentID","left")

socio\_interns=joindf.groupBy("SocioecomicStatus").agg(count("InternshipCompany"))

socio\_interns.show()

****