**IPL Final KKR vs SRH 2024 Scorecard Building Using PySpark**

!pip install pyspark findspark



import findspark

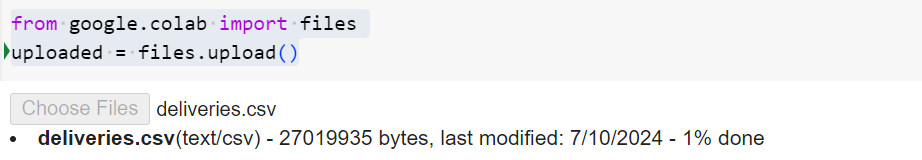
findspark.init()

from pyspark.sql import SparkSession

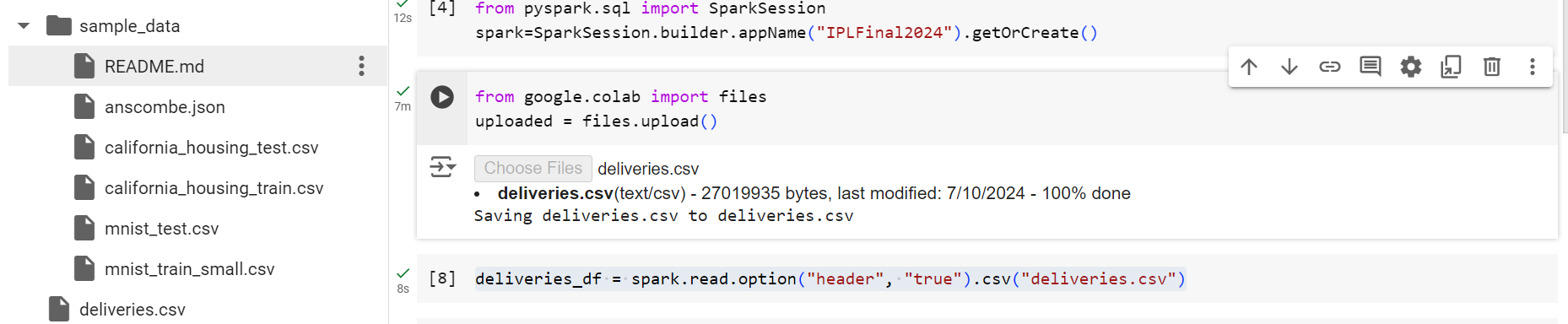
spark=SparkSession.builder.appName("IPLFinal2024").getOrCreate()

from google.colab import files

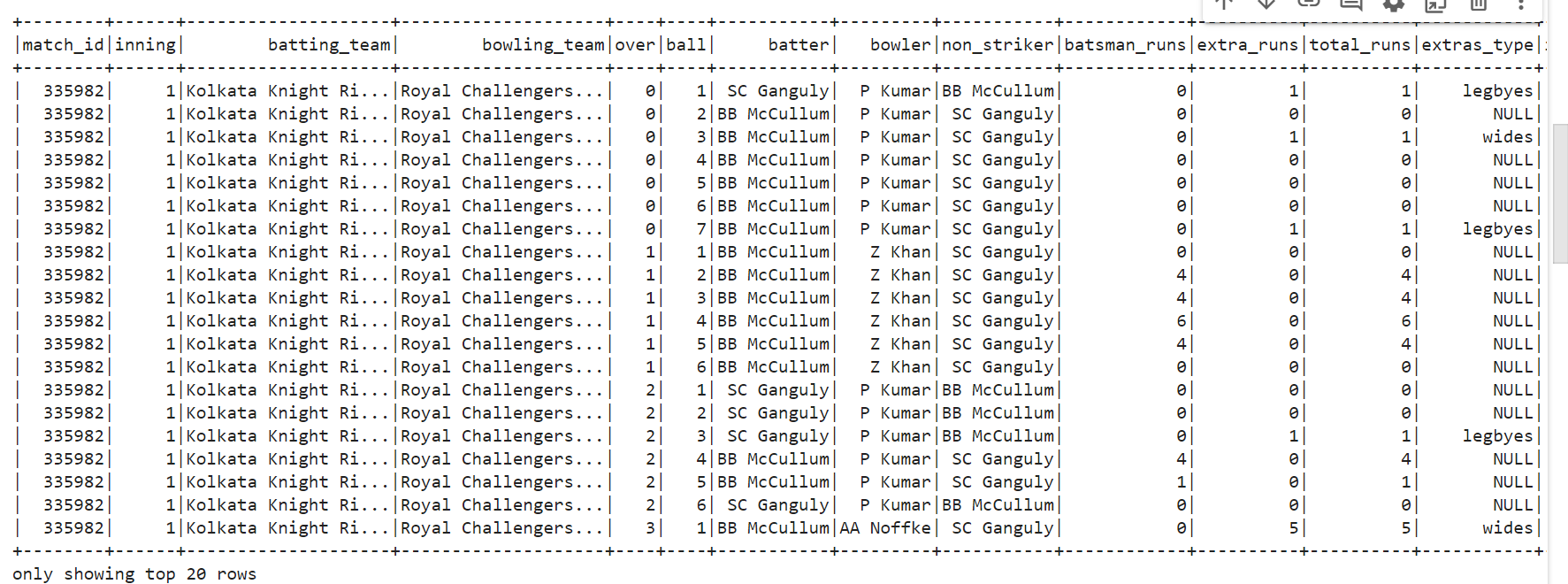
uploaded = files.upload()

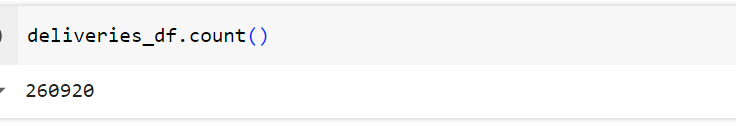


deliveries\_df = spark.read.option("header", "true").csv("deliveries.csv")



deliveries\_df.show()





deliveries\_df.show(deliveries\_df.count(),truncate=False) #df.show() Shows only top 20 rows

#truncate=False if not set it will truncate strings more than 20 chars eg. Royal Challegers ...

#Let's find final match ID

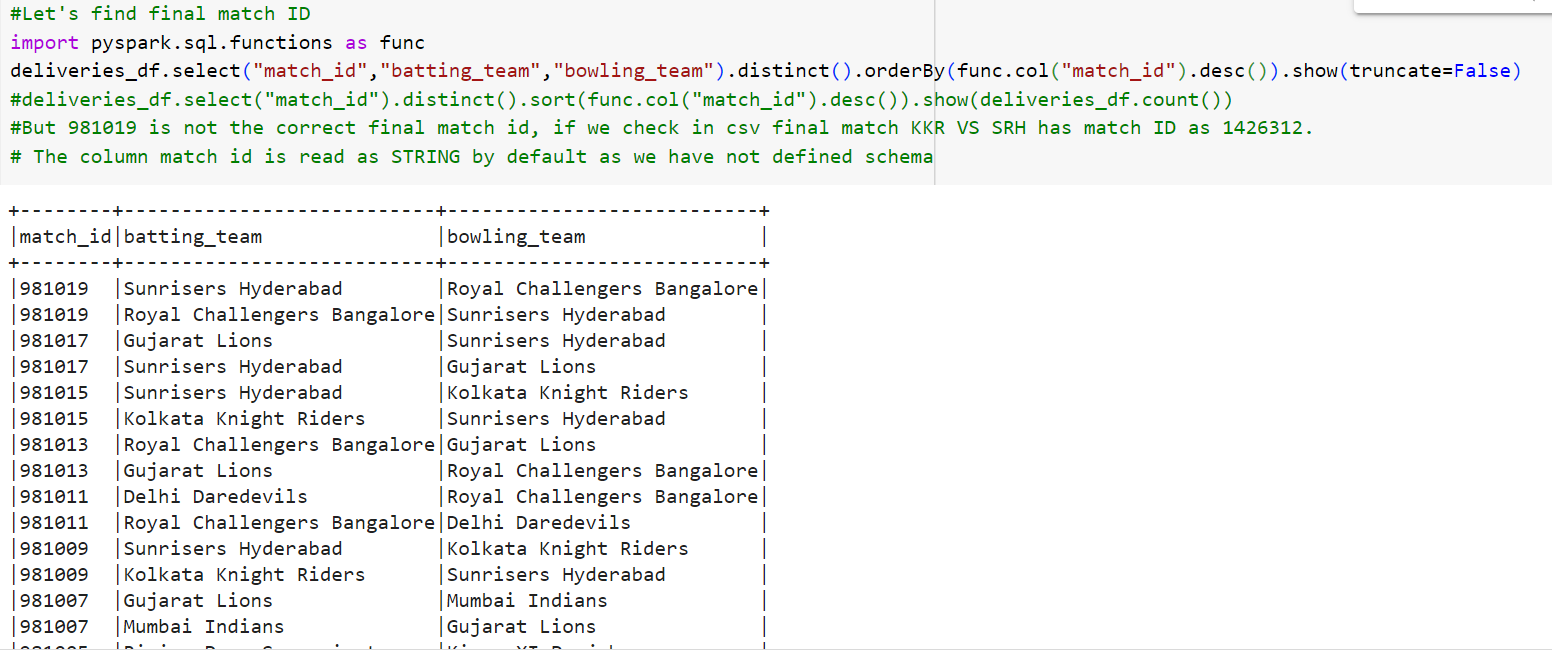
import pyspark.sql.functions as func

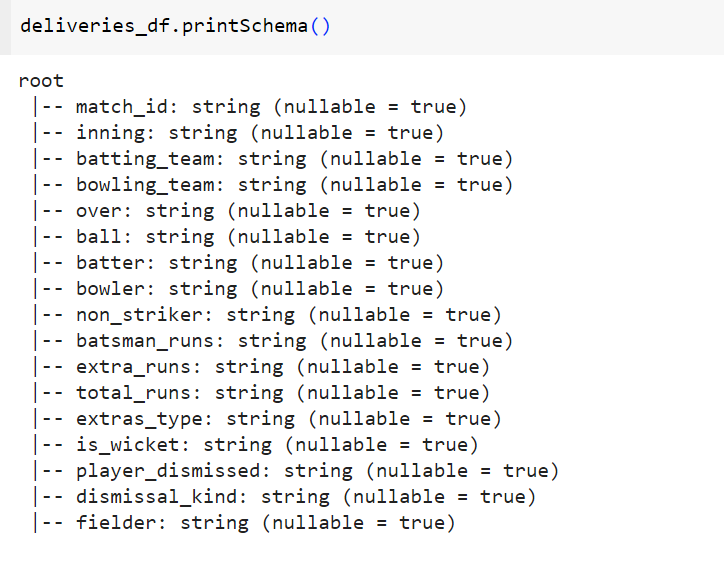
deliveries\_df.select("match\_id").distinct().orderBy(func.col("match\_id").desc()).show()

#deliveries\_df.select("match\_id").distinct().sort(func.col("match\_id").desc()).show(deliveries\_df.count())

#But 981019 is not the correct final match id, if we check in csv final match KKR VS SRH has match ID as 1426312.

# The column match id is read as STRING by default as we have not defined schema





#Let's define schema

from pyspark.sql.types import StringType, StructField

from pyspark.sql.types import \*

int\_col\_list=["match\_id","inning","over","ball","batsman\_runs","extra\_runs","total\_runs","is\_wicket"]

fields = StructType([ StructField("match\_id", IntegerType(),nullable=True) ,  StructField("inning", IntegerType(),nullable=True),

                      StructField("batting\_team", StringType(),nullable=True),StructField("bowling\_team", StringType(),nullable=True),

                      StructField("over", IntegerType(),nullable=True),       StructField("ball", IntegerType(),nullable=True),

                      StructField("batter", StringType(),nullable=True),      StructField("bowler", StringType(),nullable=True),

                      StructField("non\_striker", IntegerType(),nullable=True), StructField("batsman\_runs", IntegerType(),nullable=True),

                      StructField("extra\_runs", IntegerType(),nullable=True), StructField("total\_runs", IntegerType(),nullable=True),

                      StructField("extras\_type", StringType(),nullable=True), StructField("is\_wicket", IntegerType(),nullable=True),

                      StructField("player\_dismissed", StringType(),nullable=True),StructField("dismissal\_kind", StringType(),nullable=True),

                      StructField("fielder", StringType(),nullable=True)

])

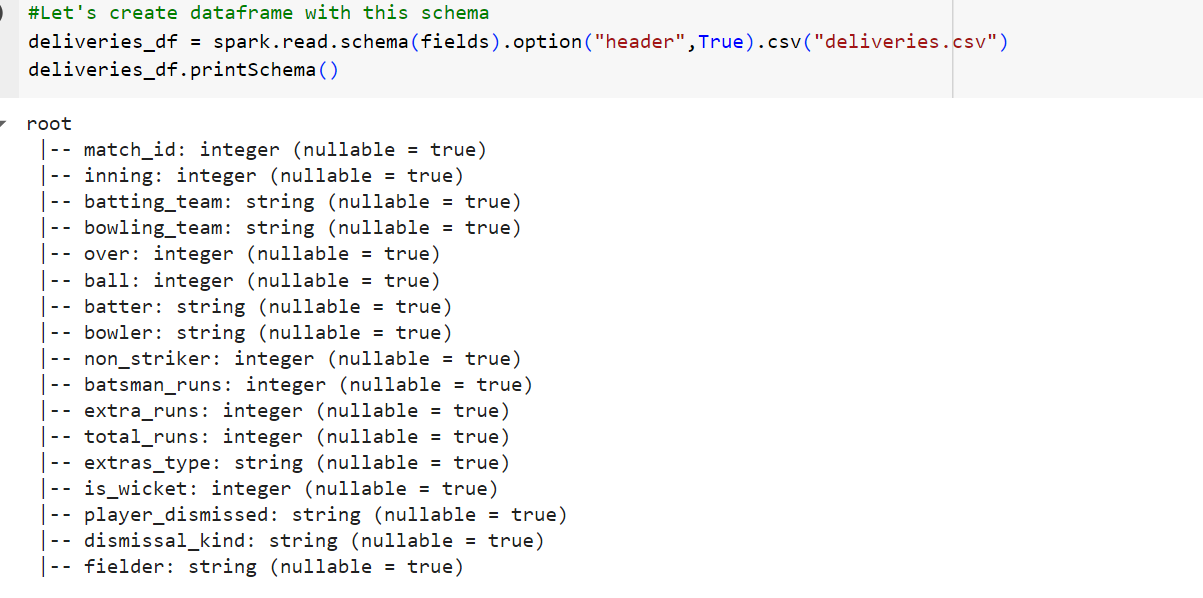
fields



#Let's create dataframe with this schema

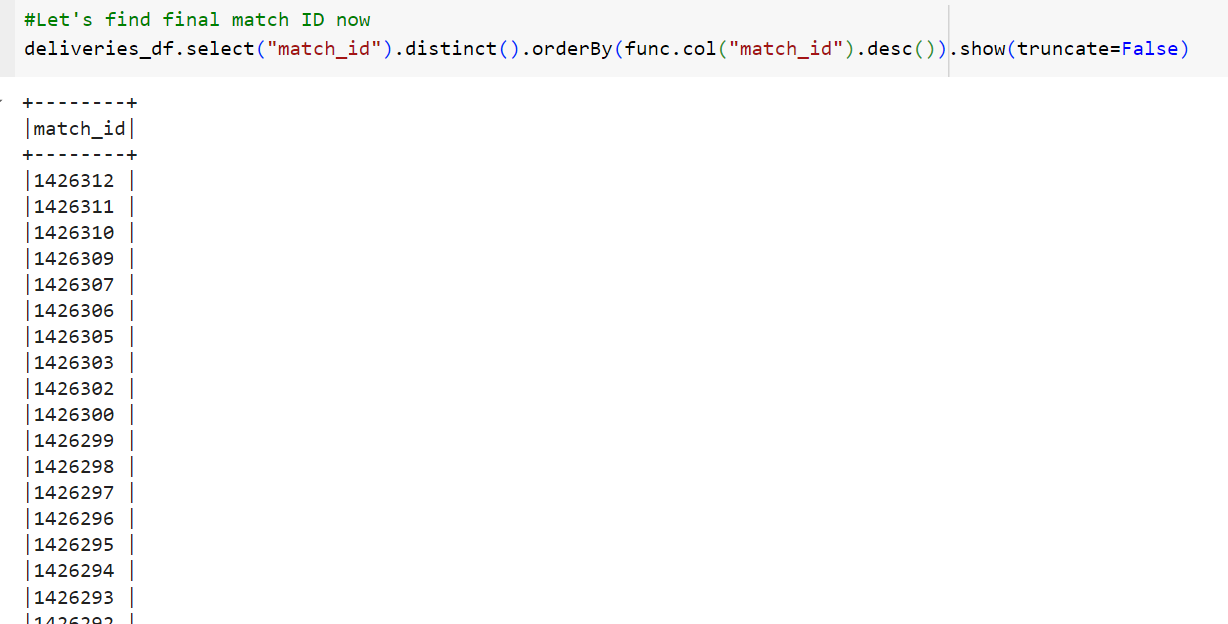
deliveries\_df = spark.read.schema(fields).option("header",True).csv("deliveries.csv")

deliveries\_df.printSchema()



#Let's find final match ID now

deliveries\_df.select("match\_id").distinct().orderBy(func.col("match\_id").desc()).show(truncate=False)

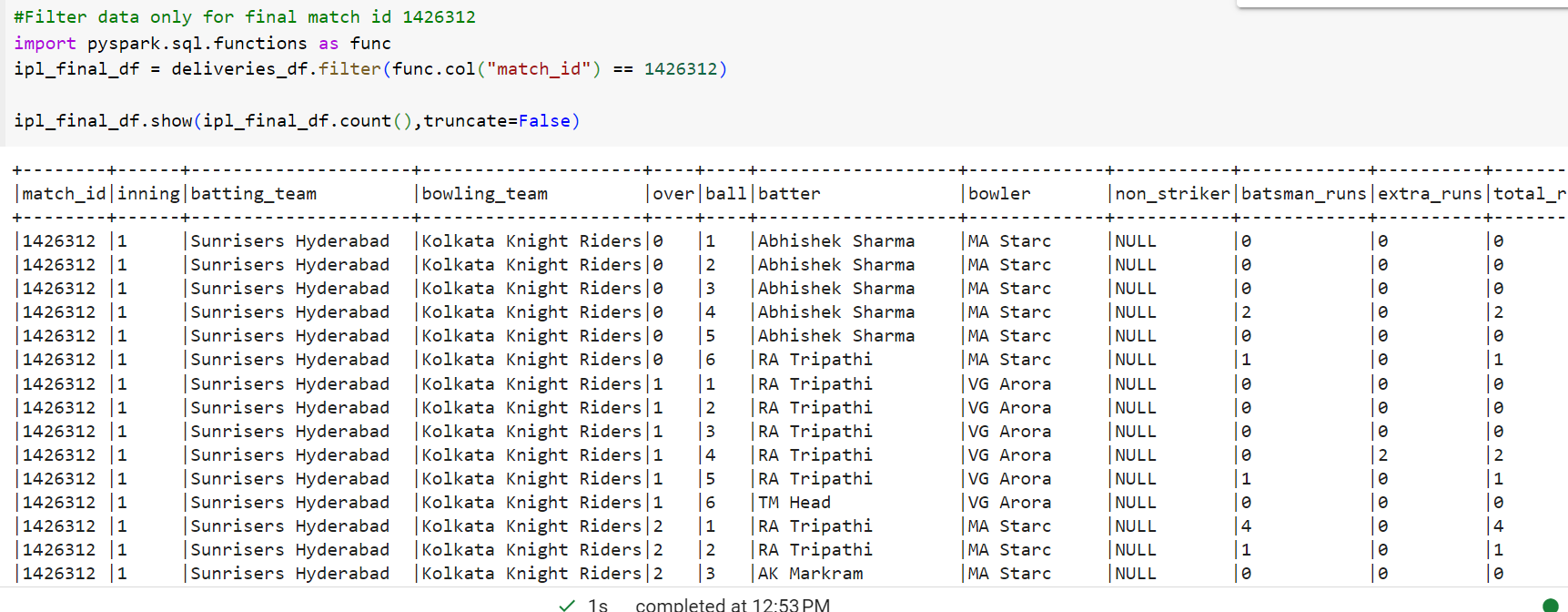


#Filter data only for final match id 1426312

import pyspark.sql.functions as func

ipl\_final\_df = deliveries\_df.filter(func.col("match\_id") == 1426312)

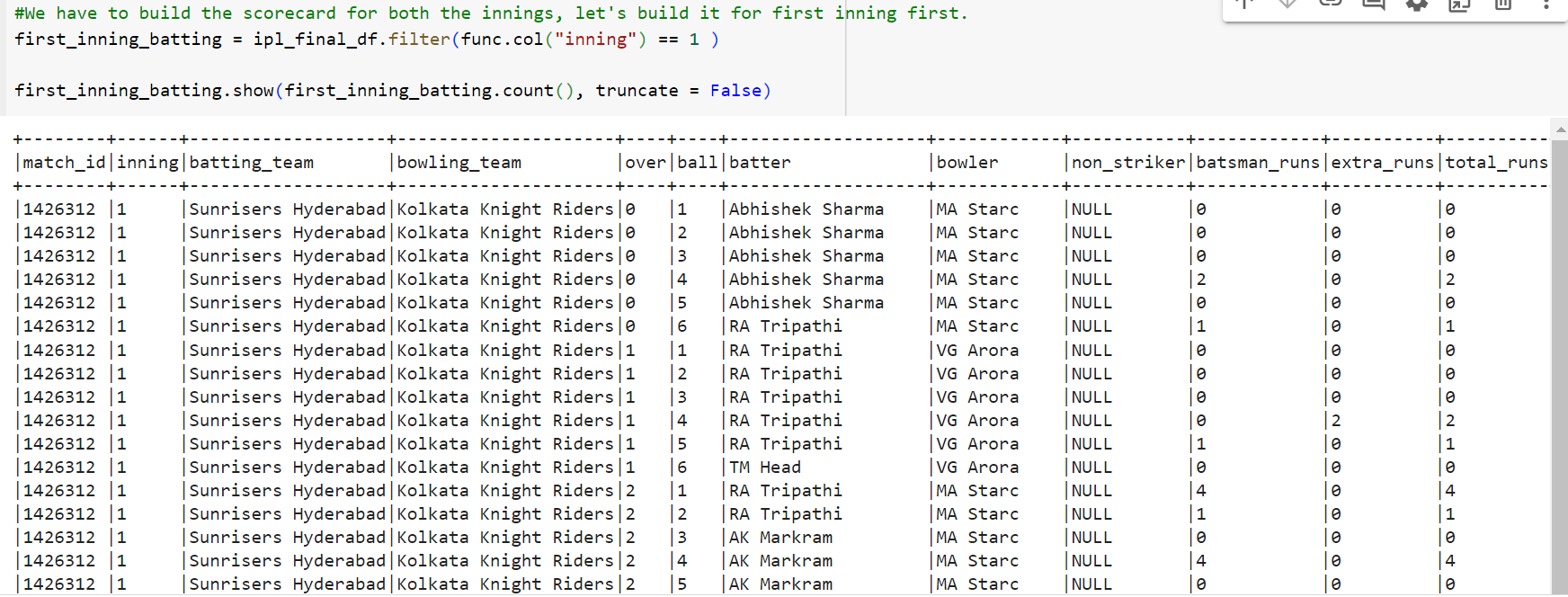
ipl\_final\_df.show(ipl\_final\_df.count(),truncate=False)



#We have to build the scorecard for both the innings, let's build it for first inning first.

first\_inning\_batting = ipl\_final\_df.filter(func.col("inning") == 1 )

first\_inning\_batting.show(first\_inning\_batting.count(), truncate = False)



#Find run scored by each batsman

batsman\_run\_df = first\_inning\_batting.filter(func.col("extras\_type").isNull() ).groupBy(func.col("batter")).agg(func.sum("batsman\_runs").alias("R"))

batsman\_run\_df.show()



#Find balls faced by each batsman

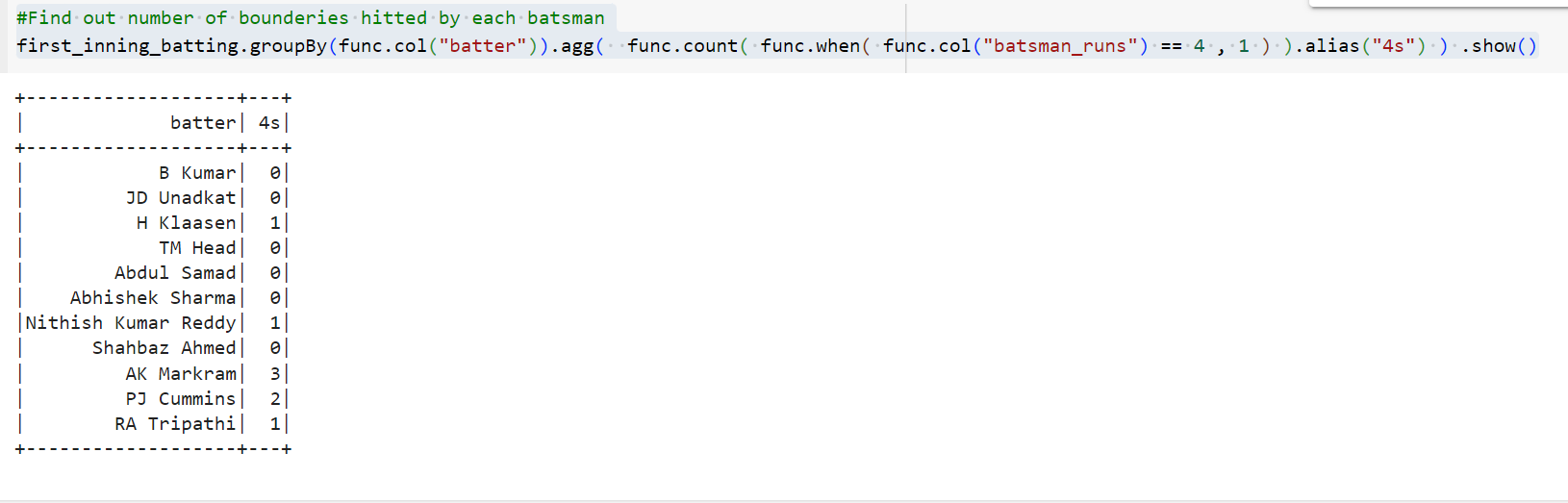
ball\_faced\_df = first\_inning\_batting.groupBy(func.col("batter")).agg(func.count("ball").alias("balls\_faced"))

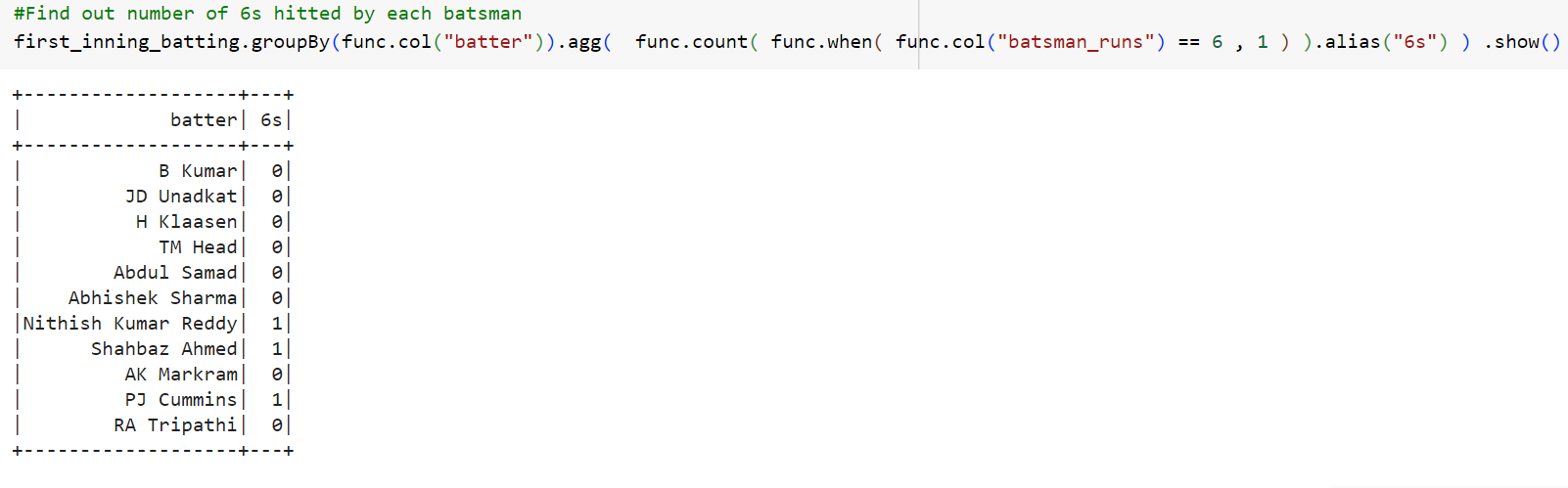
ball\_faced\_df.show()



#Find out number of bounderies hitted by each batsman

first\_inning\_batting.groupBy(func.col("batter")).agg(  func.count( func.when( func.col("batsman\_runs") == 4 , 1 ) ).alias("4s") ) .show()





#Let's build final scorecard now

Scorecard\_df = first\_inning\_batting.filter(func.col("extras\_type").isNull() ).groupBy(func.col("batter")).agg(func.sum("batsman\_runs").alias("R"),

          func.count("ball").alias("balls"),

          func.count( func.when( func.col("batsman\_runs") == 4 , 1 ) ).alias("4s"),

          func.count( func.when( func.col("batsman\_runs") == 6 , 1 ) ).alias("6s"),

          func.round(func.sum(func.col("batsman\_runs"))\*100 / func.count("ball") , 2).alias("S/R")

        )

#Let's find batsman order

batsman\_order = first\_inning\_batting.withColumn("over-ball", ( func.concat( func.col("over"),func.lit("."),func.col("ball") ) )

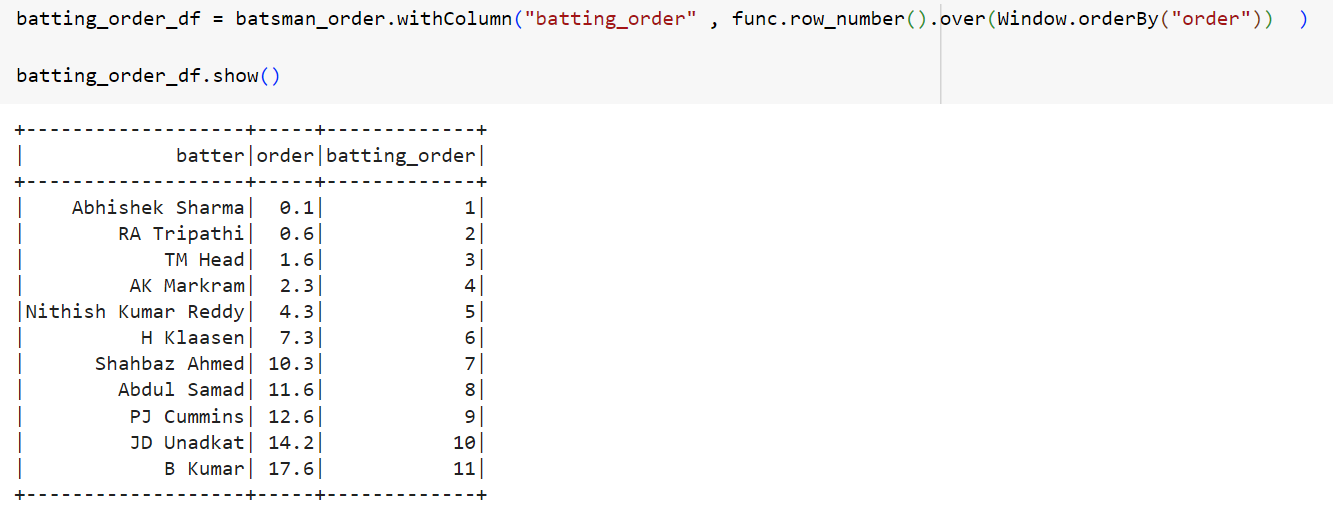
.cast('Float')).groupBy("batter").agg(func.min("over-ball").alias("order")).orderBy("order")

batsman\_order.show()



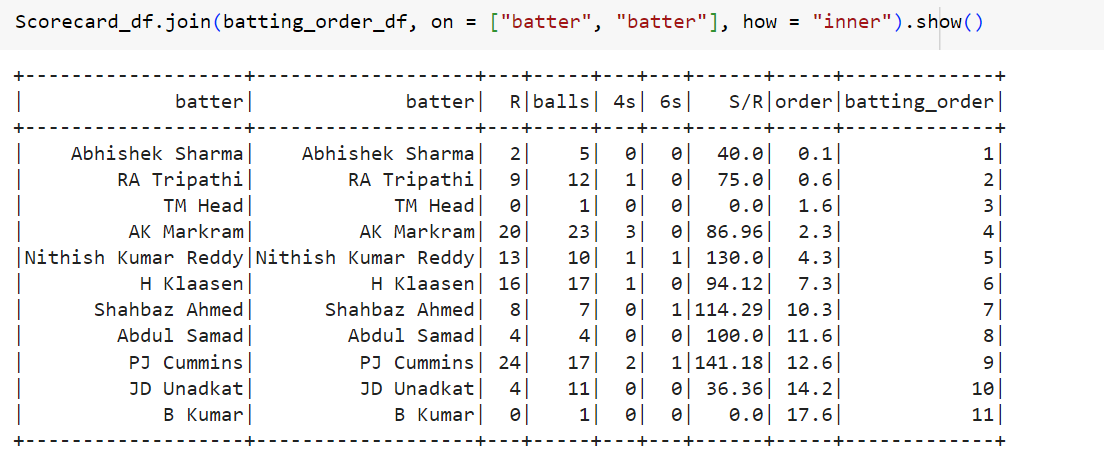
batting\_order\_df = batsman\_order.withColumn("batting\_order" , func.row\_number().over(Window.orderBy("order"))  )

batting\_order\_df.show()

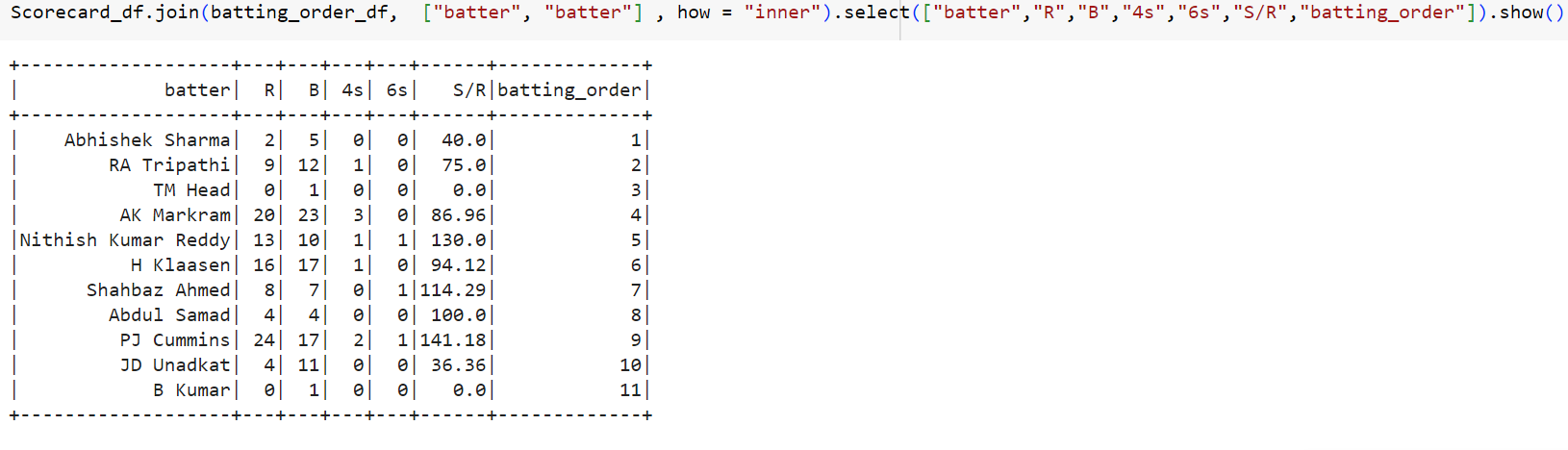




Scorecard\_df.join(batting\_order\_df, on = ["batter", "batter"], how = "inner").show()



Scorecard\_df.join(batting\_order\_df,  ["batter", "batter"] , how = "inner").select(["batter","R","B","4s","6s","S/R","batting\_order"]).show()



#Let's build bowling scorecard now

first\_inning\_bowling\_df = ipl\_final\_df.groupBy("bowler") \

.agg(

    func.count( func.when(func.coalesce(func.col("extras\_type"), func.lit("XYZ")) != "wides" ,1)).alias("balls\_bowled"), # find over using this /6 & this

    # logic func.concat(func.floor(func.count("over")/6 ), func.lit('.') ,func.count("over")%6  ) .alias("O") ,

    func.sum(func.when(func.col("extras\_type").isNull(), func.col("batsman\_runs"))  # add wide to this

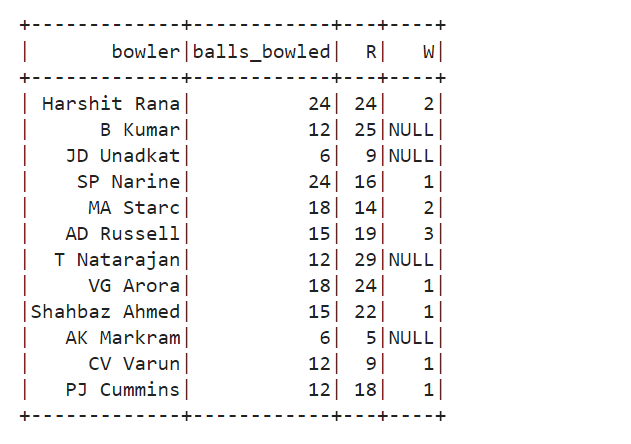
                 .when(func.col("extras\_type") == "wides", func.col("extra\_runs"))

           ).alias("R"),

    func.sum(func.when(func.col("is\_wicket") == 1, 1)).alias("W")

    )

first\_inning\_bowling\_df.show()



first\_inning\_scorecard\_df = \

first\_inning\_bowling\_df.select( "bowler",

    func.concat(func.floor(func.col("balls\_bowled")/6 ), func.lit('.') ,func.col("balls\_bowled")%6  ) .alias("O"),

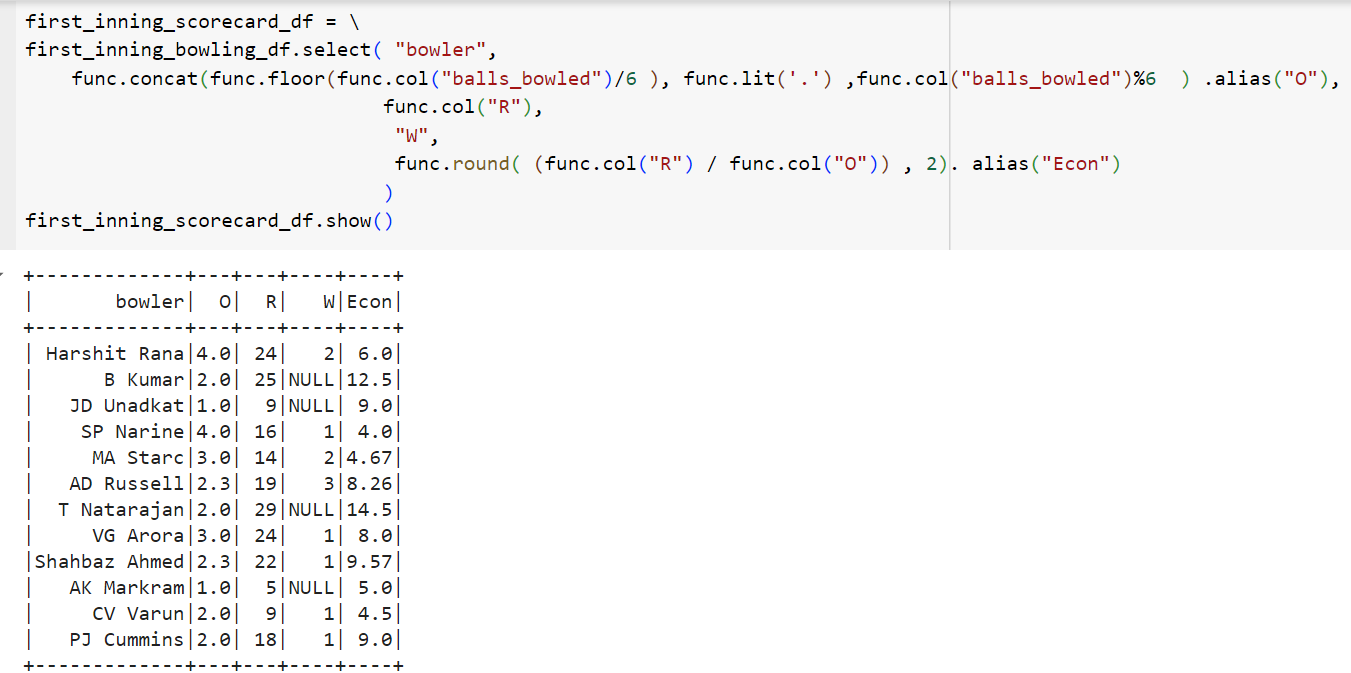
                               func.col("R"),

                                "W",

                                func.round( (func.col("R") / func.col("O")) , 2). alias("Econ")

                               )

first\_inning\_scorecard\_df.show()



maiden\_over\_df = first\_inning\_batting.groupBy("bowler","over").agg(func.sum(func.col("total\_runs")).alias("runs"),

                                                                      func.count(func.col("over")).alias("balls"))

maiden\_over\_df = maiden\_over\_df.filter((func.col("runs") == 0) & (func.col("balls") ==6 ))\

                                .groupBy("bowler").agg(func.count("bowler").alias("M"))

maiden\_over\_df.show()



first\_inning\_final\_df = first\_inning\_scorecard\_df.join(maiden\_over\_df, on = ["bowler","bowler"], how="left").fillna(value=0)

first\_inning\_final\_df.show()

