Execution of codes

-> spark-shell

-> val df = spark.read.option("header", "true").option("inferSchema", "true") .csv("/IPL/Match2.csv")

-> df.printSchema()

-> val nullCounts = df.select(df.columns.map(c => sum(col(c).isNull.cast("int")).alias(c)): \_\*)

-> nullCounts.show()

-> val matchesPerSeason = df.groupBy("Season\_Year").count()

-> matchesPerSeason.show()

-> val matchesPerCountry = df.groupBy( "Country\_Name") . count()

-> matchesPerCountry. show()

-> val matchesPerCity = df.groupBy( "City\_Name") . count()

-> matchesPerCity.show(false)

-> val topVenues = df.groupBy("Venue\_Name").count() . limit(5)

-> topVenues.show(false)

-> val team1 = df. select( "Team1")

-> val team2 = df.select("Team2")

-> val uniqueTeams = team1.union(team2).distinct()

-> uniqueTeams.show(false)

-> val tossWinnerCounts = df.groupBy( "Toss\_Winner") . count()

-> val maxTossWinner = tossWinnerCounts.orderBy(desc("count")).limit(5)

-> maxTossWinner. show(false)

-> val tossAndMatchWinnerCount = df.filter(col("Toss\_Winner")).count()

-> println(s"Matches where toss winner also won the match: $tossAndMatchWinnerCount")

-> val tossWinnerCounts = df.groupBy("Toss\_Winner").count()

-> val maxTossWinner = tossWinnerCounts.orderBy(desc("count")).limit(5)

-> maxTossWinner.show(false)

-> val matchWinnerCounts = df.groupBy("match\_winner").count()

-> val topFiveMatchWinners = matchWinnerCounts.limit(5)

-> topFiveMatchWinners.show()

-> val manOfTheMatchCounts = df.groupBy("ManOfMach").count()

-> val topFivePlayers = manOfTheMatchCounts.orderBy(desc("count")).limit(5)

-> topFivePlayers.show()

-> val defendingWins = df.filter(col("Win\_Type") === "runs")

-> val teamDefendingWins = defendingWins.groupBy("match\_winner").count()

-> val mostWinsTeam = teamDefendingWins.orderBy(desc("count")).first()

-> println(mostWinsTeam)

-> val mostWinsCount = mostwinsTeam.getLong(1)

-> val topFiveTeams = teamDefendingWins.orderBy(desc("count")).limit(5)

-> topFiveTeams.show()

-> import org.apache.spark.sql.functions.avg

-> val avgRunsMargin = df.filter(col("Win\_Type") === "runs").agg(avg("Win\_Margin"))

-> avgRunsMargin.show()

-> val matchesWonFielding = df.filter(col("Toss\_Name") === "field").groupBy("match\_winner").count().orderBy(desc("count"))

-> matchesWonFielding.show()

-> val chasingWins = df.filter(col("Win\_Type") === "wickets")

-> val teamChasingWins = chasingWins.groupBy("match\_winner").count()

-> val mostWinsTeam = teamChasingWins.orderBy(desc("count")).first()

-> val mostWinsCount = mostWinsTeam.getLong(1)

-> val topFiveTeams = teamChasingWins.orderBy(desc("count")).limit(5)

-> topFiveTeams.show()

-> val avgWicketsMargin = df.filter(col("Win\_Type") === "wickets").agg(avg("Win\_Margin"))

-> avgWicketsMargin.show()