eda-with-pyspark-and-sparksql

August 19, 2024

[0]:
```
%fs ls dbfs:/FileStore/tables/ipl/
```

[0]:
```
from pyspark.sql import SparkSession
spark=SparkSession.builder.appName("IPL").getOrCreate()
```

[0]:
```
from pyspark.sql.types import StructField ,StructType ,StringType␣
 ↪,DateType,BooleanType ,IntegerType,DecimalType
```

[0]:
```
Ball_schema = StructType([
    StructField("match_id", IntegerType(), True),
    StructField("over_id", IntegerType(), True),
    StructField("ball_id", IntegerType(), True),
    StructField("innings_no", IntegerType(), True),
    StructField("team_batting", StringType(), True),
    StructField("team_bowling", StringType(), True),
    StructField("striker_batting_position", IntegerType(), True),
    StructField("extra_type", StringType(), True),
    StructField("runs_scored", IntegerType(), True),
    StructField("extra_runs", IntegerType(), True),
    StructField("wides", IntegerType(), True),
    StructField("legbyes", IntegerType(), True),
    StructField("byes", IntegerType(), True),
    StructField("noballs", IntegerType(), True),
    StructField("penalty", IntegerType(), True),
    StructField("bowler_extras", IntegerType(), True),
    StructField("out_type", StringType(), True),
    StructField("caught", BooleanType(), True),
    StructField("bowled", BooleanType(), True),
    StructField("run_out", BooleanType(), True),
    StructField("lbw", BooleanType(), True),
    StructField("retired_hurt", BooleanType(), True),
    StructField("stumped", BooleanType(), True),
    StructField("caught_and_bowled", BooleanType(), True),
    StructField("hit_wicket", BooleanType(), True),
    StructField("obstructingfeild", BooleanType(), True),
    StructField("bowler_wicket", BooleanType(), True),
    StructField("match_date", DateType(), True),
```

```
    StructField("season", IntegerType(), True),
    StructField("striker", IntegerType(), True),
    StructField("non_striker", IntegerType(), True),
    StructField("bowler", IntegerType(), True),
    StructField("player_out", IntegerType(), True),
    StructField("fielders", IntegerType(), True),
    StructField("striker_match_sk", IntegerType(), True),
    StructField("strikersk", IntegerType(), True),
    StructField("nonstriker_match_sk", IntegerType(), True),
    StructField("nonstriker_sk", IntegerType(), True),
    StructField("fielder_match_sk", IntegerType(), True),
    StructField("fielder_sk", IntegerType(), True),
    StructField("bowler_match_sk", IntegerType(), True),
    StructField("bowler_sk", IntegerType(), True),
    StructField("playerout_match_sk", IntegerType(), True),
    StructField("battingteam_sk", IntegerType(), True),
    StructField("bowlingteam_sk", IntegerType(), True),
    StructField("keeper_catch", BooleanType(), True),
    StructField("player_out_sk", IntegerType(), True),
    StructField("matchdatesk", DateType(), True)
])
```

[0]:
```
ball_by_ball=spark.read.format('csv') \
    .option('header',True) \
    .schema(Ball_schema) \
    .load('dbfs:/FileStore/tables/ipl/Ball_By_Ball.csv')
```

[0]:
```
match_schema = StructType([
    StructField("match_sk", IntegerType(), True),
    StructField("match_id", IntegerType(), True),
    StructField("team1", StringType(), True),
    StructField("team2", StringType(), True),
    StructField("match_date", DateType(), True),
    StructField("season_year", IntegerType(), True),
    StructField("venue_name", StringType(), True),
    StructField("city_name", StringType(), True),
    StructField("country_name", StringType(), True),
    StructField("toss_winner", StringType(), True),
    StructField("match_winner", StringType(), True),
    StructField("toss_name", StringType(), True),
    StructField("win_type", StringType(), True),
    StructField("outcome_type", StringType(), True),
    StructField("manofmach", StringType(), True),
    StructField("win_margin", IntegerType(), True),
    StructField("country_id", IntegerType(), True)
])
```

```
match_df = spark.read.schema(match_schema).format("csv").
  ↪option("header","true").load("dbfs:/FileStore/tables/ipl/Match.csv")
```

[0]:
```
player_schema = StructType([
    StructField("player_sk", IntegerType(), True),
    StructField("player_id", IntegerType(), True),
    StructField("player_name", StringType(), True),
    StructField("dob", DateType(), True),
    StructField("batting_hand", StringType(), True),
    StructField("bowling_skill", StringType(), True),
    StructField("country_name", StringType(), True)
])

player_df = spark.read.schema(player_schema).format("csv").
  ↪option("header","true").load("s3://ipl-data-analysis-project/Player.csv")
```

[0]:
```
player_match_schema = StructType([
    StructField("player_match_sk", IntegerType(), True),
    StructField("playermatch_key", DecimalType(), True),
    StructField("match_id", IntegerType(), True),
    StructField("player_id", IntegerType(), True),
    StructField("player_name", StringType(), True),
    StructField("dob", DateType(), True),
    StructField("batting_hand", StringType(), True),
    StructField("bowling_skill", StringType(), True),
    StructField("country_name", StringType(), True),
    StructField("role_desc", StringType(), True),
    StructField("player_team", StringType(), True),
    StructField("opposit_team", StringType(), True),
    StructField("season_year", IntegerType(), True),
    StructField("is_manofthematch", BooleanType(), True),
    StructField("age_as_on_match", IntegerType(), True),
    StructField("isplayers_team_won", BooleanType(), True),
    StructField("batting_status", StringType(), True),
    StructField("bowling_status", StringType(), True),
    StructField("player_captain", StringType(), True),
    StructField("opposit_captain", StringType(), True),
    StructField("player_keeper", StringType(), True),
    StructField("opposit_keeper", StringType(), True)
])

player_match_df = spark.read.schema(player_match_schema).format("csv").
  ↪option("header","true").load("s3://ipl-data-analysis-project/Player_match.
  ↪csv")
```

[0]:
```
team_schema = StructType([
    StructField("team_sk", IntegerType(), True),
```

```
    StructField("team_id", IntegerType(), True),
    StructField("team_name", StringType(), True)
])

team_df = spark.read.schema(team_schema).format("csv").option("header","true").
 ↪load("s3://ipl-data-analysis-project/Team.csv")
```

[0]:
```
from pyspark.sql.functions import col,sum,avg,round, when , row_number
from pyspark.sql.window import Window
```

[0]:
```
# Filter to include only valid deliveries (excluding extras like wides and no␣
 ↪balls for specific analyses)
ball_by_ball=ball_by_ball.filter((col("wides")==0) & (col("extra_runs")==0) )
```

[0]:
```
# Aggregation: Calculate the total and average runs scored in each match and␣
 ↪inning
#ball_by_ball.groupBy('')

aggregated_df=ball_by_ball \
    .groupBy("match_id", "innings_no") \
    .agg(
        sum("runs_scored").alias("sum"),
        round(avg("runs_scored"), 2).alias("avg")
    )
```

[0]:
```
# Window Function: Calculate running total of runs in each match for each over

ball_by_ball=ball_by_ball.withColumn(
    "running_total",
    sum("runs_scored").over(
        Window.partitionBy('match_id','innings_no','over_id').orderBy('over_id')
    )
)
```

[0]:
```
# Conditional Column: Flag for high impact balls (either a wicket or more than␣
 ↪6 runs including extras)
ball_by_ball=ball_by_ball.withColumn(
    'high impact'
    ,when ((col('runs_scored')>6) | (col('bowler_wicket')==True),True).
 ↪otherwise(False)
)
```

[0]:
```
from pyspark.sql.functions import year, month, dayofmonth, when
# Extracting year, month, and day from the match date for more detailed␣
 ↪time-based analysis
match_df = match_df.withColumn("year", year("match_date"))
```

```
match_df = match_df.withColumn("month", month("match_date"))
match_df = match_df.withColumn("day", dayofmonth("match_date"))
```

[0]:
```
# High margin win: categorizing win margins into 'high', 'medium', and 'low'


match_df.withColumn(
    'win_margin',
    when(col("win_margin")>=100,"high").
    when((col("win_margin")>=50)  & (col("win_margin")<100),"medium").
    otherwise('low')

)
```

Out[16]: DataFrame[match_sk: int, match_id: int, team1: string, team2: string, match_date: date, season_year: int, venue_name: string, city_name: string, country_name: string, toss_winner: string, match_winner: string, toss_name: string, win_type: string, outcome_type: string, manofmach: string, win_margin: string, country_id: int, year: int, month: int, day: int]

[0]:
```
ball_by_ball.createOrReplaceTempView("ball_by_ball")
match_df.createOrReplaceTempView("match")
player_df.createOrReplaceTempView("player")
player_match_df.createOrReplaceTempView("player_match")
team_df.createOrReplaceTempView("team")
```

[0]:
```
%sql
select * from ball_by_ball
```

[0]:
```
#top scoring batsmen per_season
%sql
select
      P.PLAYER_NAME,
      M.SEASON_YEAR,
      SUM(B.RUNS_SCORED)
from ball_by_ball b
join match m on b.match_id = M.MATCH_ID
JOIN PLAYER_MATCH PM ON M.MATCH_ID = PM.MATCH_ID  AND  b.striker = pm.player_id
JOIN PLAYER P ON P.PLAYER_ID=PM.PLAYER_ID
group by  P.PLAYER_NAME,M.SEASON_YEAR
ORDER BY 2,3 DESC
```

[0]:
```
#economical
# bowlers in powerplay
%sql
SELECT
p.player_name,
```

```
round(AVG(b.runs_scored),2) AS avg_runs_per_ball,
COUNT(b.bowler_wicket) AS total_wickets
FROM ball_by_ball b
JOIN player_match pm ON b.match_id = pm.match_id AND b.bowler = pm.player_id
JOIN player p ON pm.player_id = p.player_id
WHERE b.over_id <= 6
GROUP BY p.player_name
HAVING COUNT(*) >= 1
ORDER BY avg_runs_per_ball, total_wickets DESC
```

[0]:
```
#toss impact individual matches
%sql
SELECT m.match_id, m.toss_winner, m.toss_name, m.match_winner,
       CASE WHEN m.toss_winner = m.match_winner THEN 'Won' ELSE 'Lost' END AS␣
 ↪match_outcome
FROM match m
WHERE m.toss_name IS NOT NULL
ORDER BY m.match_id
```