Exam 1 Submission Form

Overview

Your Name	Aarti Anandkumar Mehra
Your SU Email	aamehra@syr.edu

Please see https://github.com/mafudge/ist769sp23midterm for specific exam information and instructions.

Instructions

Each question is worth 6 points.

NOTE: If you cannot figure out the answer to the question, I suggest writing simpler code and use that as your answer. This way you can complete the next question in the exam. It is better to have running code that is incorrect than code that will not run.

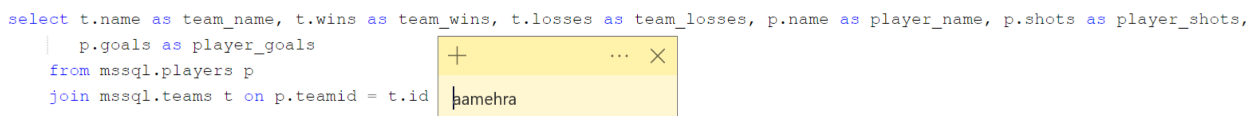
For the highest possible marks, for each question, include:

1. The TEXT of the code you wrote.
2. A CLEAR screenshot of your code with your netid in the screenshot. (Only screenshot the region, not the entire window!)
3. A CLEAR screenshot of the output of your code with your netid in the screenshot. (Only screenshot the region, not the entire window!)
4. If know your answer is incorrect, explain what you tried/omitted/did not get correct, by adding comments to your code/commenting out code that does not run. This should appear in your text and screenshot.

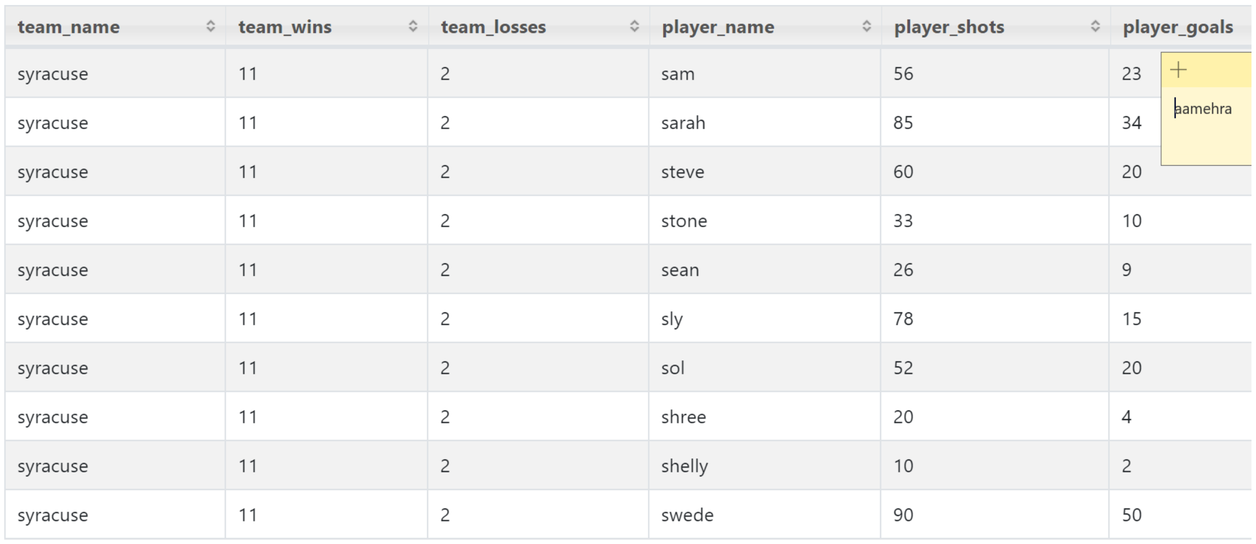
Questions

1. Write a drill SQL query to list the team and player data. Specifically display team name, team wins, team losses player name, player shots and player goals.

select t.name as team_name, t.wins as team_wins, t.losses as team_losses, p.name as player_name, p.shots as player_shots, p.goals as player_goals
    from mssql.players p
    join mssql.teams t on p.teamid = t.id



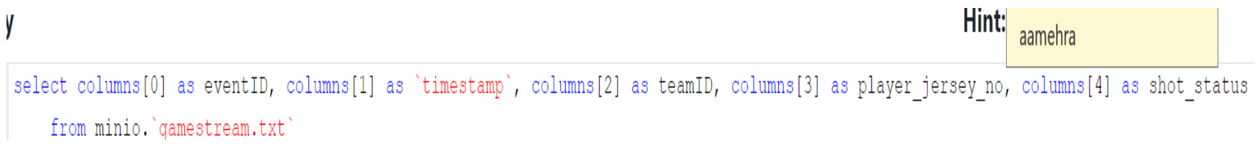
```
select t.name as team_name, t.wins as team_wins, t.losses as team_losses, p.name as player_name, p.shots as player_shots,
    p.goals as player_goals
  from mssql.players p
  join mssql.teams t on p.teamid = t.id
```



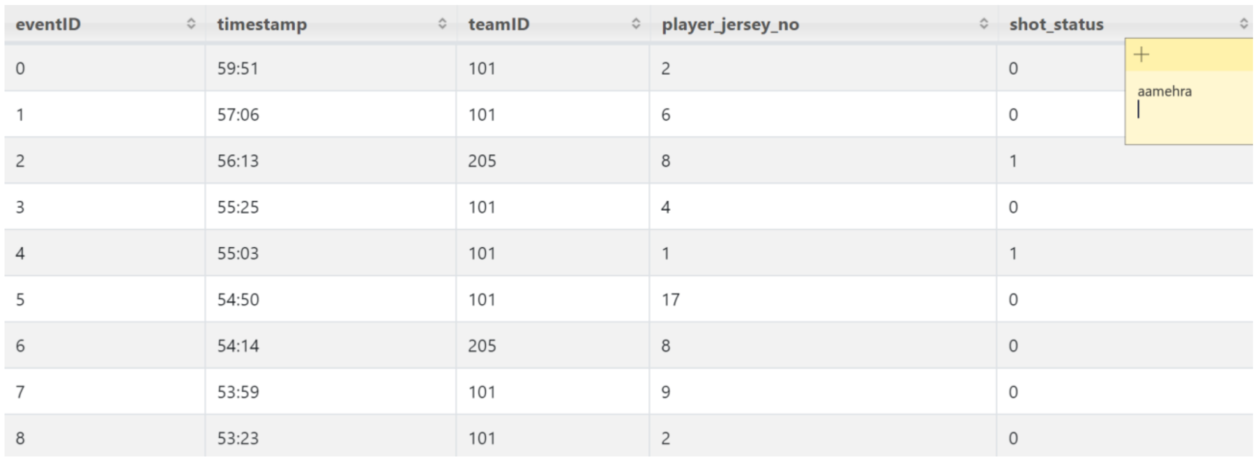
team_name	team_wins	team_losses	player_name	player_shots	player_goals
syracuse	11	2	sam	56	23
syracuse	11	2	sarah	85	34
syracuse	11	2	steve	60	20
syracuse	11	2	stone	33	10
syracuse	11	2	sean	26	9
syracuse	11	2	sly	78	15
syracuse	11	2	sol	52	20
syracuse	11	2	shree	20	4
syracuse	11	2	shelly	10	2
syracuse	11	2	swede	90	50

2. Write a drill SQL query to display the gamestream. Label each of the columns in the gamestream with their appropriate columns names from the data dictionary.

   select columns[0] as eventID, columns[1] as `timestamp`, columns[2] as teamID, columns[3] as player_jersey_no, columns[4] as shot_status
      from minio.`gamestream.txt`



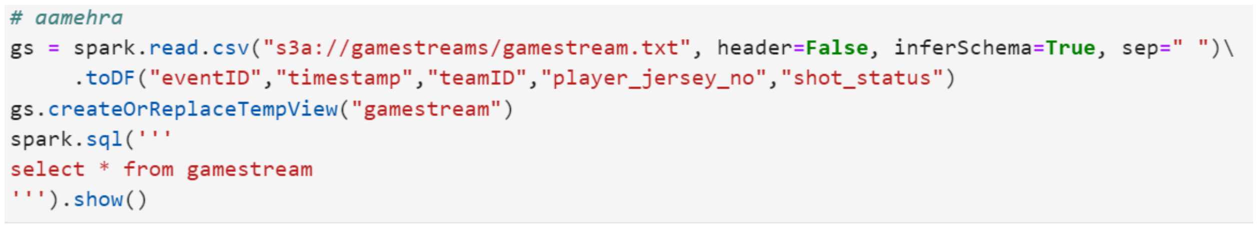
```
select columns[0] as eventID, columns[1] as `timestamp`, columns[2] as teamID, columns[3] as player_jersey_no, columns[4] as shot_status
    from minio.`gamestream.txt`
```



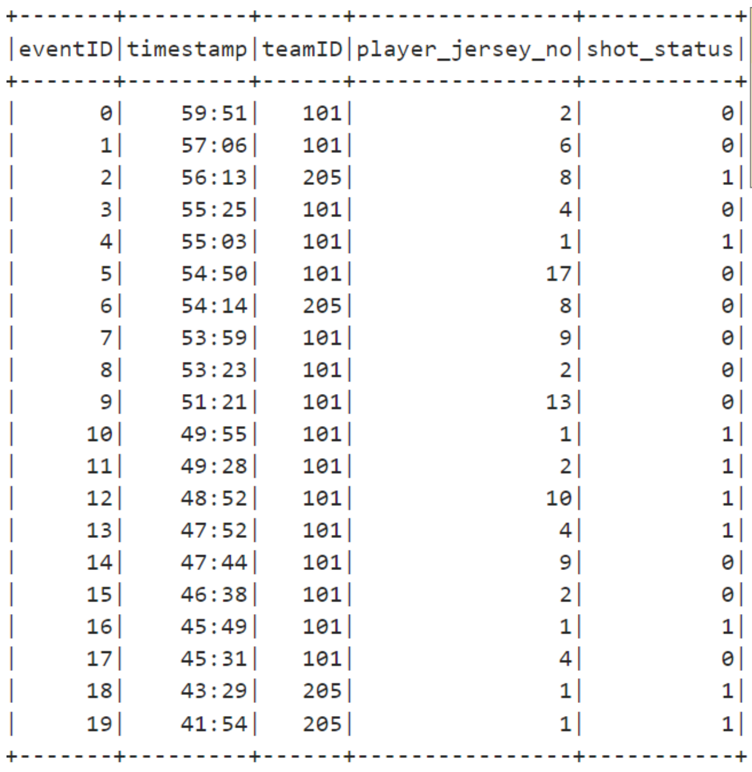
eventID	timestamp	teamID	player_jersey_no	shot_status
0	59:51	101	2	0
1	57:06	101	6	0
2	56:13	205	8	1
3	55:25	101	4	0
4	55:03	101	1	1
5	54:50	101	17	0
6	54:14	205	8	0
7	53:59	101	9	0
8	53:23	101	2	0

3. Write pyspark code (in SQL or DataFrame API) to display the gamestream. Label each of the columns in the gamestream with their appropriate columns names from the data dictionary.

   gs = spark.read.csv("s3a://gamestreams/gamestream.txt", header=False, inferSchema=True, sep=" ")\
   .toDF("eventID","timestamp","teamID","player_jersey_no","shot_status")
   gs.createOrReplaceTempView("gamestream")
   spark.sql('''
   select * from gamestream
   ''').show()



```
# aamehra
gs = spark.read.csv("s3a://gamestreams/gamestream.txt", header=False, inferSchema=True, sep=" ")\
     .toDF("eventID","timestamp","teamID","player_jersey_no","shot_status")
gs.createOrReplaceTempView("gamestream")
spark.sql('''
select * from gamestream
''').show()
```



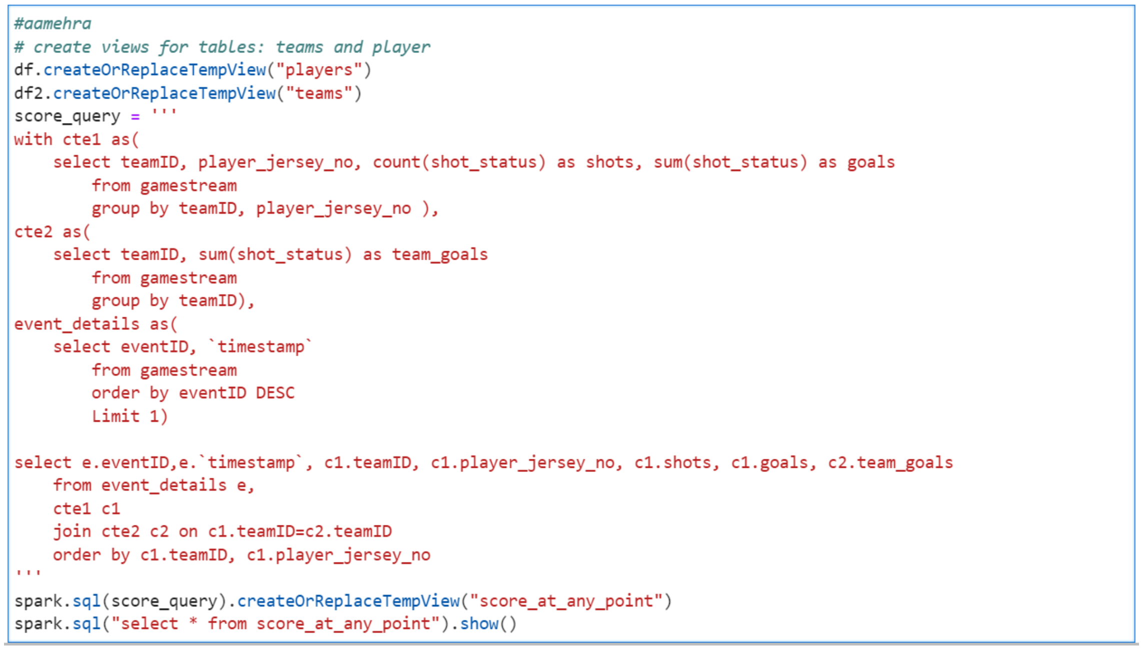
```
+-------+---------+------+---------------+-----------+
|eventID|timestamp|teamID|player_jersey_no|shot_status|
+-------+---------+------+---------------+-----------+
|      0|    59:51|   101|              2|          0|
|      1|    57:06|   101|              6|          0|
|      2|    56:13|   205|              8|          1|
|      3|    55:25|   101|              4|          0|
|      4|    55:03|   101|              1|          1|
|      5|    54:50|   101|             17|          0|
|      6|    54:14|   205|              8|          0|
|      7|    53:59|   101|              9|          0|
|      8|    53:23|   101|              2|          0|
|      9|    51:21|   101|             13|          0|
|     10|    49:55|   101|              1|          1|
|     11|    49:28|   101|              2|          1|
|     12|    48:52|   101|             10|          1|
|     13|    47:52|   101|              4|          1|
|     14|    47:44|   101|              9|          0|
|     15|    46:38|   101|              2|          0|
|     16|    45:49|   101|              1|          1|
|     17|    45:31|   101|              4|          0|
|     18|    43:29|   205|              1|          1|
|     19|    41:54|   205|              1|          1|
+-------+---------+------+---------------+-----------+
```

aamehra

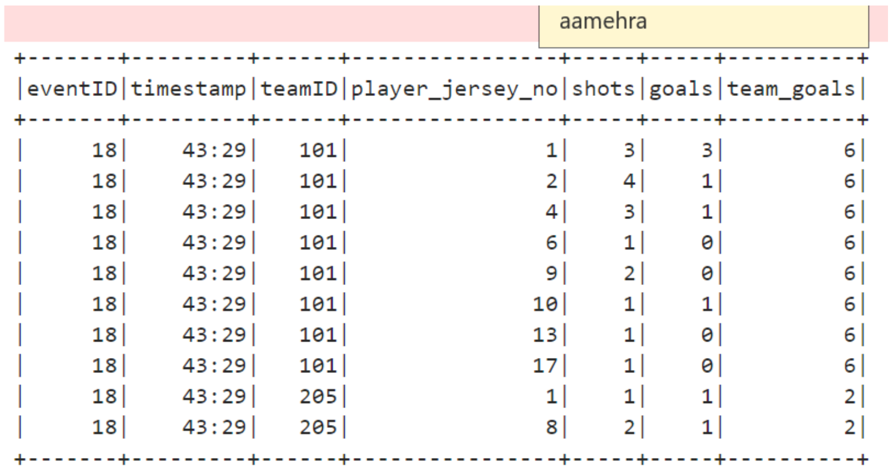
4. Write pyspark code (in SQL or DataFrame API) to group the gamestream by team/player id adding up the shots and goals.
   - Include the team score.
   - Include the latest event id and the timestamp for that event id.

```
df.createOrReplaceTempView("players")
df2.createOrReplaceTempView("teams")
score_query = '''
with cte1 as(
   select teamID, player_jersey_no, count(shot_status) as shots, sum(shot_status)
as goals
      from gamestream
      group by teamID, player_jersey_no ),
cte2 as(
   select teamID, sum(shot_status) as team_goals
      from gamestream
      group by teamID),
event_details as(
   select eventID, `timestamp`
      from gamestream
      order by eventID DESC
```

```
        Limit 1)

select e.eventID,e.`timestamp`, c1.teamID, c1.player_jersey_no, c1.shots, c1.goals,
c2.team_goals
    from event_details e,
    cte1 c1
    join cte2 c2 on c1.teamID=c2.teamID
    order by c1.teamID, c1.player_jersey_no
'''
spark.sql(score_query).createOrReplaceTempView("score_at_any_point")
spark.sql("select * from score_at_any_point").show()
```



```
#aamehra
# create views for tables: teams and player
df.createOrReplaceTempView("players")
df2.createOrReplaceTempView("teams")
score_query = '''
with cte1 as(
    select teamID, player_jersey_no, count(shot_status) as shots, sum(shot_status) as goals
        from gamestream
        group by teamID, player_jersey_no ),
cte2 as(
    select teamID, sum(shot_status) as team_goals
        from gamestream
        group by teamID),
event_details as(
    select eventID, `timestamp`
        from gamestream
        order by eventID DESC
        Limit 1)

select e.eventID,e.`timestamp`, c1.teamID, c1.player_jersey_no, c1.shots, c1.goals, c2.team_goals
    from event_details e,
    cte1 c1
    join cte2 c2 on c1.teamID=c2.teamID
    order by c1.teamID, c1.player_jersey_no
'''
spark.sql(score_query).createOrReplaceTempView("score_at_any_point")
spark.sql("select * from score_at_any_point").show()
```

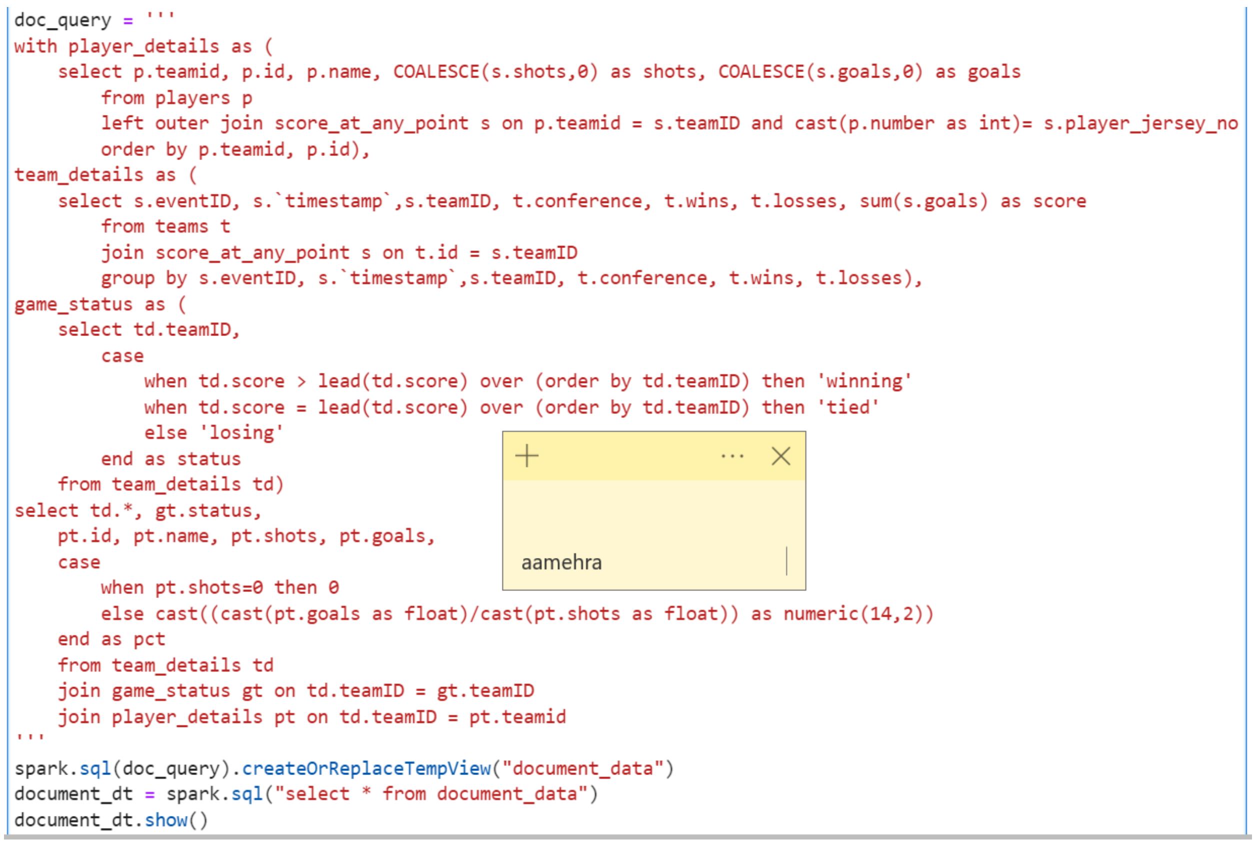


aamehra

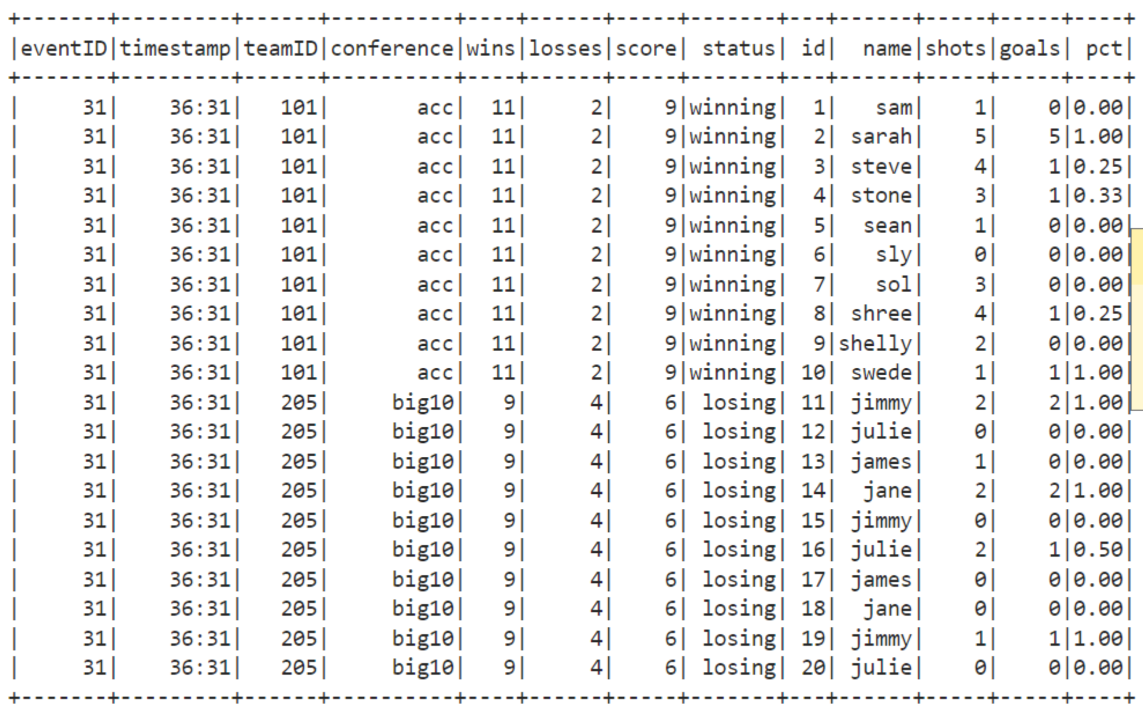
```
+-------+---------+------+----------------+-----+-----+----------+
|eventID|timestamp|teamID|player_jersey_no|shots|goals|team_goals|
+-------+---------+------+----------------+-----+-----+----------+
|     18|    43:29|   101|               1|    3|    3|         6|
|     18|    43:29|   101|               2|    4|    1|         6|
|     18|    43:29|   101|               4|    3|    1|         6|
|     18|    43:29|   101|               6|    1|    0|         6|
|     18|    43:29|   101|               9|    2|    0|         6|
|     18|    43:29|   101|              10|    1|    1|         6|
|     18|    43:29|   101|              13|    1|    0|         6|
|     18|    43:29|   101|              17|    1|    0|         6|
|     18|    43:29|   205|               1|    1|    1|         2|
|     18|    43:29|   205|               8|    2|    1|         2|
+-------+---------+------+----------------+-----+-----+----------+
```

5. Write pyspark code (in SQL or DataFrame API) to join the output from question 4 with the player and team reference data `mssql` so that you have the data necessary for the box score.

```
doc_query = '''
with player_details as (
   select p.teamid, p.id, p.name, COALESCE(s.shots,0) as shots,
COALESCE(s.goals,0) as goals
      from players p
      left outer join score_at_any_point s on p.teamid = s.teamID and
cast(p.number as int)= s.player_jersey_no
      order by p.teamid, p.id),
team_details as (
   select s.eventID, s.`timestamp`,s.teamID, t.conference, t.wins, t.losses,
sum(s.goals) as score
      from teams t
      join score_at_any_point s on t.id = s.teamID
      group by s.eventID, s.`timestamp`,s.teamID, t.conference, t.wins, t.losses),
game_status as (
   select td.teamID,
      case
         when td.score > lead(td.score) over (order by td.teamID) then 'winning'
         when td.score = lead(td.score) over (order by td.teamID) then 'tied'
         else 'losing'
      end as status
   from team_details td)
select td.*, gt.status,
   pt.id, pt.name, pt.shots, pt.goals,
   case
      when pt.shots=0 then 0
      else cast((cast(pt.goals as float)/cast(pt.shots as float)) as numeric(14,2))
   end as pct
   from team_details td
   join game_status gt on td.teamID = gt.teamID
   join player_details pt on td.teamID = pt.teamid
'''
spark.sql(doc_query).createOrReplaceTempView("document_data")
document_dt = spark.sql("select * from document_data")
document_dt.show()
```



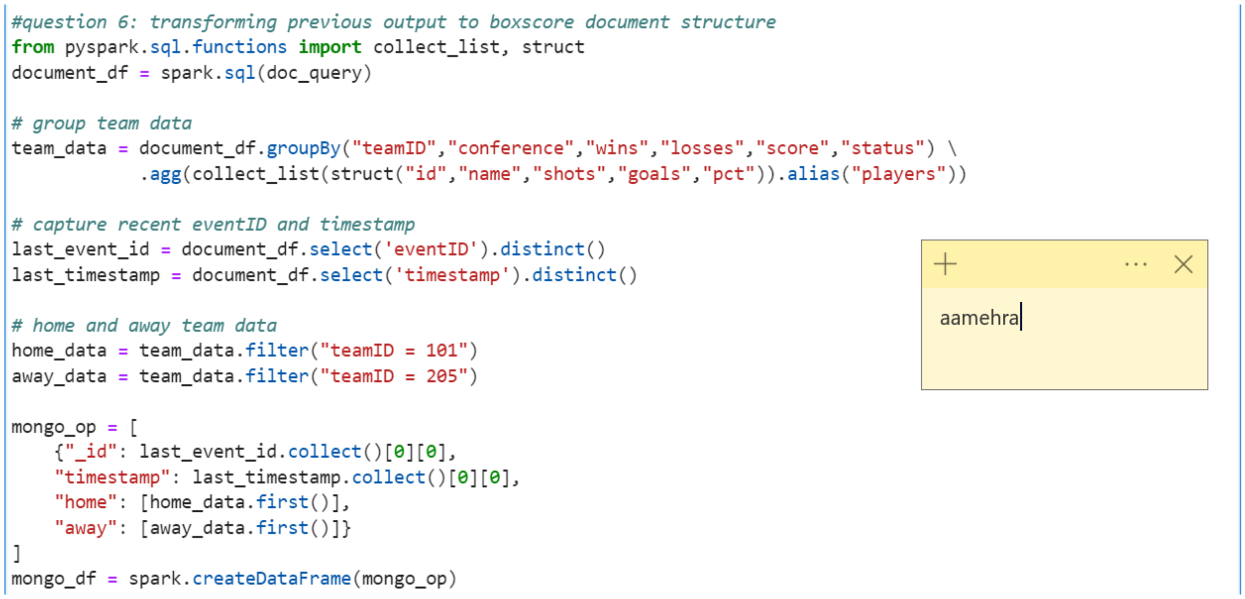
```
doc_query = '''
with player_details as (
    select p.teamid, p.id, p.name, COALESCE(s.shots,0) as shots, COALESCE(s.goals,0) as goals
        from players p
        left outer join score_at_any_point s on p.teamid = s.teamID and cast(p.number as int)= s.player_jersey_no
        order by p.teamid, p.id),
team_details as (
    select s.eventID, s.`timestamp`,s.teamID, t.conference, t.wins, t.losses, sum(s.goals) as score
        from teams t
        join score_at_any_point s on t.id = s.teamID
        group by s.eventID, s.`timestamp`,s.teamID, t.conference, t.wins, t.losses),
game_status as (
    select td.teamID,
        case
            when td.score > lead(td.score) over (order by td.teamID) then 'winning'
            when td.score = lead(td.score) over (order by td.teamID) then 'tied'
            else 'losing'
        end as status
    from team_details td)
select td.*, gt.status,
    pt.id, pt.name, pt.shots, pt.goals,
    case
        when pt.shots=0 then 0
        else cast((cast(pt.goals as float)/cast(pt.shots as float)) as numeric(14,2))
    end as pct
    from team_details td
    join game_status gt on td.teamID = gt.teamID
    join player_details pt on td.teamID = pt.teamid
'''
spark.sql(doc_query).createOrReplaceTempView("document_data")
document_dt = spark.sql("select * from document_data")
document_dt.show()
```



```
+-------+---------+------+----------+----+------+-----+-------+---+------+-----+-----+----+
|eventID|timestamp|teamID|conference|wins|losses|score| status| id|  name|shots|goals| pct|
+-------+---------+------+----------+----+------+-----+-------+---+------+-----+-----+----+
|     31|    36:31|   101|       acc|  11|     2|    9|winning|  1|   sam|    1|    0|0.00|
|     31|    36:31|   101|       acc|  11|     2|    9|winning|  2| sarah|    5|    5|1.00|
|     31|    36:31|   101|       acc|  11|     2|    9|winning|  3| steve|    4|    1|0.25|
|     31|    36:31|   101|       acc|  11|     2|    9|winning|  4| stone|    3|    1|0.33|
|     31|    36:31|   101|       acc|  11|     2|    9|winning|  5|  sean|    1|    0|0.00|
|     31|    36:31|   101|       acc|  11|     2|    9|winning|  6|   sly|    0|    0|0.00|
|     31|    36:31|   101|       acc|  11|     2|    9|winning|  7|   sol|    3|    0|0.00|
|     31|    36:31|   101|       acc|  11|     2|    9|winning|  8| shree|    4|    1|0.25|
|     31|    36:31|   101|       acc|  11|     2|    9|winning|  9|shelly|    2|    0|0.00|
|     31|    36:31|   101|       acc|  11|     2|    9|winning| 10| swede|    1|    1|1.00|
|     31|    36:31|   205|     big10|   9|     4|    6| losing| 11| jimmy|    2|    2|1.00|
|     31|    36:31|   205|     big10|   9|     4|    6| losing| 12| julie|    0|    0|0.00|
|     31|    36:31|   205|     big10|   9|     4|    6| losing| 13| james|    1|    0|0.00|
|     31|    36:31|   205|     big10|   9|     4|    6| losing| 14|  jane|    2|    2|1.00|
|     31|    36:31|   205|     big10|   9|     4|    6| losing| 15| jimmy|    0|    0|0.00|
|     31|    36:31|   205|     big10|   9|     4|    6| losing| 16| julie|    2|    1|0.50|
|     31|    36:31|   205|     big10|   9|     4|    6| losing| 17| james|    0|    0|0.00|
|     31|    36:31|   205|     big10|   9|     4|    6| losing| 18|  jane|    0|    0|0.00|
|     31|    36:31|   205|     big10|   9|     4|    6| losing| 19| jimmy|    1|    1|1.00|
|     31|    36:31|   205|     big10|   9|     4|    6| losing| 20| julie|    0|    0|0.00|
+-------+---------+------+----------+----+------+-----+-------+---+------+-----+-----+----+
```

6. Write pyspark code (in SQL or DataFrame API) to transform the output from question 5 into the box score document structure shown in part 3.1.

```
from pyspark.sql.functions import collect_list, struct
document_df = spark.sql(doc_query)

# group team data
team_data =
document_df.groupBy("teamID","conference","wins","losses","score","status") \
          .agg(collect_list(struct("id","name","shots","goals","pct")).alias("players"))

# capture recent eventID and timestamp
last_event_id = document_df.select('eventID').distinct()
last_timestamp = document_df.select('timestamp').distinct()

# home and away team data
home_data = team_data.filter("teamID = 101")
away_data = team_data.filter("teamID = 205")

mongo_op = [
   {"_id": last_event_id.collect()[0][0],
   "timestamp": last_timestamp.collect()[0][0],
   "home": [home_data.first()],
   "away": [away_data.first()]}
]
mongo_df = spark.createDataFrame(mongo_op)
```

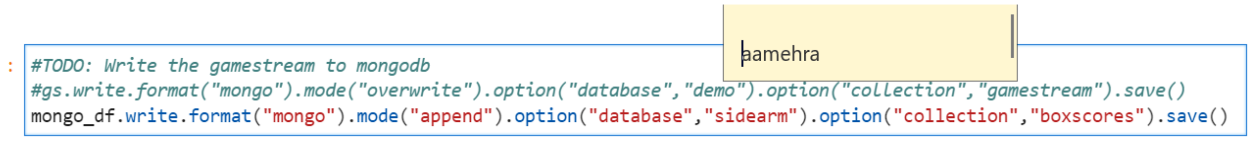


```
#question 6: transforming previous output to boxscore document structure
from pyspark.sql.functions import collect_list, struct
document_df = spark.sql(doc_query)

# group team data
team_data = document_df.groupBy("teamID","conference","wins","losses","score","status") \
            .agg(collect_list(struct("id","name","shots","goals","pct")).alias("players"))

# capture recent eventID and timestamp
last_event_id = document_df.select('eventID').distinct()
last_timestamp = document_df.select('timestamp').distinct()

# home and away team data
home_data = team_data.filter("teamID = 101")
away_data = team_data.filter("teamID = 205")

mongo_op = [
    {"_id": last_event_id.collect()[0][0],
    "timestamp": last_timestamp.collect()[0][0],
    "home": [home_data.first()],
    "away": [away_data.first()]}
]
mongo_df = spark.createDataFrame(mongo_op)
```

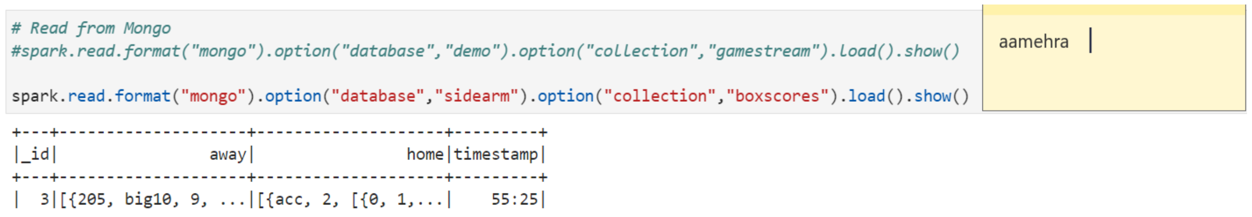
aamehra

7. Write pyspark code (in SQL or DataFrame API) to write the box score completed in question 6 to the `mongo.sidearm.boxscores` collection.

   mongo_df.write.format("mongo").mode("append").option("database","sidearm").option("collection","boxscores").save()



```
#TODO: Write the gamestream to mongodb
#gs.write.format("mongo").mode("overwrite").option("database","demo").option("collection","gamestream").save()
mongo_df.write.format("mongo").mode("append").option("database","sidearm").option("collection","boxscores").save()
```



```
# Read from Mongo
#spark.read.format("mongo").option("database","demo").option("collection","gamestream").load().show()

spark.read.format("mongo").option("database","sidearm").option("collection","boxscores").load().show()
```

```
+---+--------------------+--------------------+---------+
|_id|                away|                home|timestamp|
+---+--------------------+--------------------+---------+
|  3|[{205, big10, 9, ...|[{acc, 2, [{0, 1,...|    55:25|
```

8. Combine parts 4-7 into a single pyspark script that will run the entire process of creating the box score document. Make sure to run this a couple of times while the game stream is going on.

```
# create views for tables: teams and player
df.createOrReplaceTempView("players")
df2.createOrReplaceTempView("teams")

# question 4: grouping gamestream
score_query = '''
with cte1 as(
   select teamID, player_jersey_no, count(shot_status) as shots, sum(shot_status)
as goals
      from gamestream
      group by teamID, player_jersey_no ),
cte2 as(
   select teamID, sum(shot_status) as team_goals
      from gamestream
      group by teamID),
event_details as(
   select eventID, `timestamp`
      from gamestream
      order by eventID DESC
```

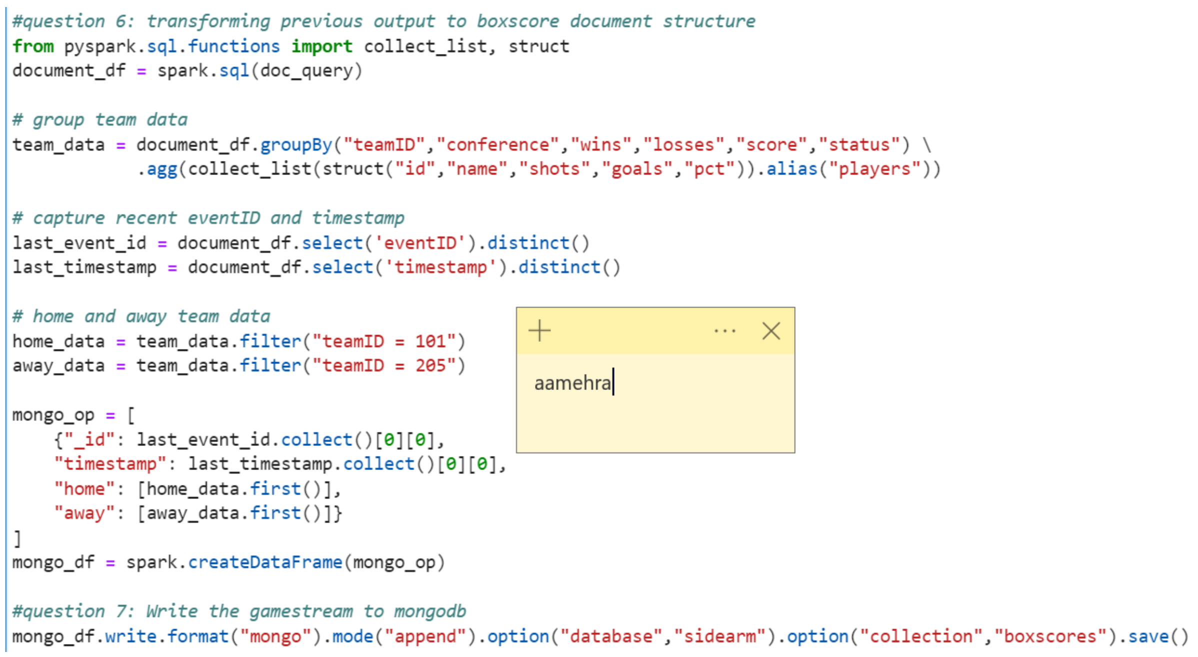
```
        Limit 1)

select e.eventID,e.`timestamp`, c1.teamID, c1.player_jersey_no, c1.shots, c1.goals, 
c2.team_goals
    from event_details e,
    cte1 c1
    join cte2 c2 on c1.teamID=c2.teamID
    order by c1.teamID, c1.player_jersey_no
'''
spark.sql(score_query).createOrReplaceTempView("score_at_any_point")
#spark.sql("select * from score_at_any_point").show()

#question 5: creation of data for mongodb boxscores collection
doc_query = '''
with player_details as (
    select p.teamid, p.id, p.name, COALESCE(s.shots,0) as shots, 
COALESCE(s.goals,0) as goals
        from players p
        left outer join score_at_any_point s on p.teamid = s.teamID and 
cast(p.number as int)= s.player_jersey_no
        order by p.teamid, p.id),
team_details as (
    select s.eventID, s.`timestamp`,s.teamID, t.conference, t.wins, t.losses, 
sum(s.goals) as score
        from teams t
        join score_at_any_point s on t.id = s.teamID
        group by s.eventID, s.`timestamp`,s.teamID, t.conference, t.wins, t.losses),
game_status as (
    select td.teamID,
        case
            when td.score > lead(td.score) over (order by td.teamID) then 'winning'
            when td.score = lead(td.score) over (order by td.teamID) then 'tied'
            else 'losing'
        end as status
    from team_details td)
select td.*, gt.status,
    pt.id, pt.name, pt.shots, pt.goals,
    case
        when pt.shots=0 then 0
        else cast((cast(pt.goals as float)/cast(pt.shots as float)) as numeric(14,2))
    end as pct
    from team_details td
    join game_status gt on td.teamID = gt.teamID
    join player_details pt on td.teamID = pt.teamid
'''
spark.sql(score_query).createOrReplaceTempView("document_data")
document_dt = spark.sql("select * from document_data")
#document_dt.show()

#question 6: transforming previous output to boxscore document structure
from pyspark.sql.functions import collect_list, struct
document_df = spark.sql(doc_query)

# group team data
team_data =
document_df.groupBy("teamID","conference","wins","losses","score","status") \
            .agg(collect_list(struct("id","name","shots","goals","pct")).alias("players"))

# capture recent eventID and timestamp
last_event_id = document_df.select('eventID').distinct()
last_timestamp = document_df.select('timestamp').distinct()

# home and away team data
home_data = team_data.filter("teamID = 101")
away_data = team_data.filter("teamID = 205")

mongo_op = [
    {"_id": last_event_id.collect()[0][0],
    "timestamp": last_timestamp.collect()[0][0],
    "home": [home_data.first()],
    "away": [away_data.first()]}
]
mongo_df = spark.createDataFrame(mongo_op)

#question 7: Write the gamestream to mongodb
mongo_df.write.format("mongo").mode("append").option("database","sidearm").o
ption("collection","boxscores").save()
```

```
# create views for tables: teams and player
df.createOrReplaceTempView("players")
df2.createOrReplaceTempView("teams")

# question 4: grouping gamestream
score_query = '''
with cte1 as(
    select teamID, player_jersey_no, count(shot_status) as shots, sum(shot_status) as goals
        from gamestream
        group by teamID, player_jersey_no ),
cte2 as(
    select teamID, sum(shot_status) as team_goals
        from gamestream
        group by teamID),
event_details as(
    select eventID, `timestamp`
        from gamestream
        order by eventID DESC
        Limit 1)

select e.eventID,e.`timestamp`, c1.teamID, c1.player_jersey_no, c1.shots, c1.goals, c2.team_goals
    from event_details e,
    cte1 c1
    join cte2 c2 on c1.teamID=c2.teamID
    order by c1.teamID, c1.player_jersey_no
'''
spark.sql(score_query).createOrReplaceTempView("score_at_any_point")
#spark.sql("select * from score_at_any_point").show()
```

aamehra

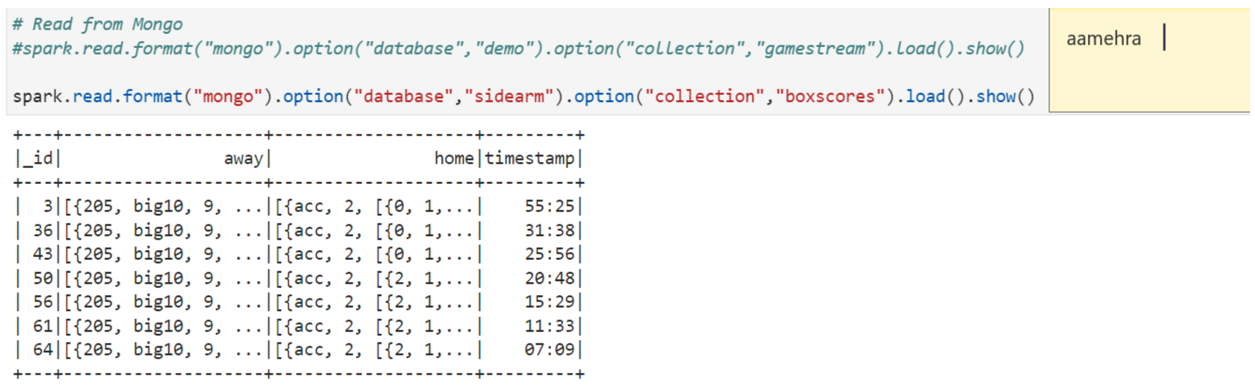
```
#question 5: creation of data for mongodb boxscores collection
doc_query = '''
with player_details as (
    select p.teamid, p.id, p.name, COALESCE(s.shots,0) as shots, COALESCE(s.goals,0) as goals
        from players p
        left outer join score_at_any_point s on p.teamid = s.teamID and cast(p.number as int)= s.player_jersey_no
        order by p.teamid, p.id),
team_details as (
    select s.eventID, s.`timestamp`,s.teamID, t.conference, t.wins, t.losses, sum(s.goals) as score
        from teams t
        join score_at_any_point s on t.id = s.teamID
        group by s.eventID, s.`timestamp`,s.teamID, t.conference, t.wins, t.losses),
game_status as (
    select td.teamID,
        case
            when td.score > lead(td.score) over (order by td.teamID) then 'winning'
            when td.score = lead(td.score) over (order by td.teamID) then 'tied'
            else 'losing'
        end as status
    from team_details td)
select td.*, gt.status,
    pt.id, pt.name, pt.shots, pt.goals,
    case
        when pt.shots=0 then 0
        else cast((cast(pt.goals as float)/cast(pt.shots as float)) as numeric(14,2))
    end as pct
    from team_details td
    join game_status gt on td.teamID = gt.teamID
    join player_details pt on td.teamID = pt.teamid
'''
spark.sql(doc_query).createOrReplaceTempView("document_data")
document_dt = spark.sql("select * from document_data")
```

aamehra



```
#question 6: transforming previous output to boxscore document structure
from pyspark.sql.functions import collect_list, struct
document_df = spark.sql(doc_query)

# group team data
team_data = document_df.groupBy("teamID","conference","wins","losses","score","status") \
            .agg(collect_list(struct("id","name","shots","goals","pct")).alias("players"))

# capture recent eventID and timestamp
last_event_id = document_df.select('eventID').distinct()
last_timestamp = document_df.select('timestamp').distinct()

# home and away team data
home_data = team_data.filter("teamID = 101")
away_data = team_data.filter("teamID = 205")

mongo_op = [
    {"_id": last_event_id.collect()[0][0],
    "timestamp": last_timestamp.collect()[0][0],
    "home": [home_data.first()],
    "away": [away_data.first()]}
]
mongo_df = spark.createDataFrame(mongo_op)

#question 7: Write the gamestream to mongodb
mongo_df.write.format("mongo").mode("append").option("database","sidearm").option("collection","boxscores").save()
```

aamehra



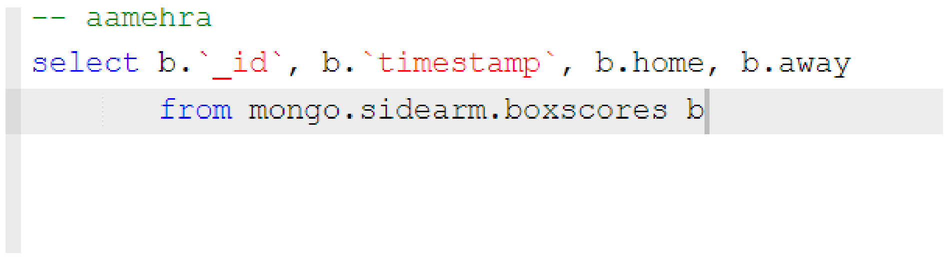
```
# Read from Mongo
#spark.read.format("mongo").option("database","demo").option("collection","gamestream").load().show()

spark.read.format("mongo").option("database","sidearm").option("collection","boxscores").load().show()
```

aamehra

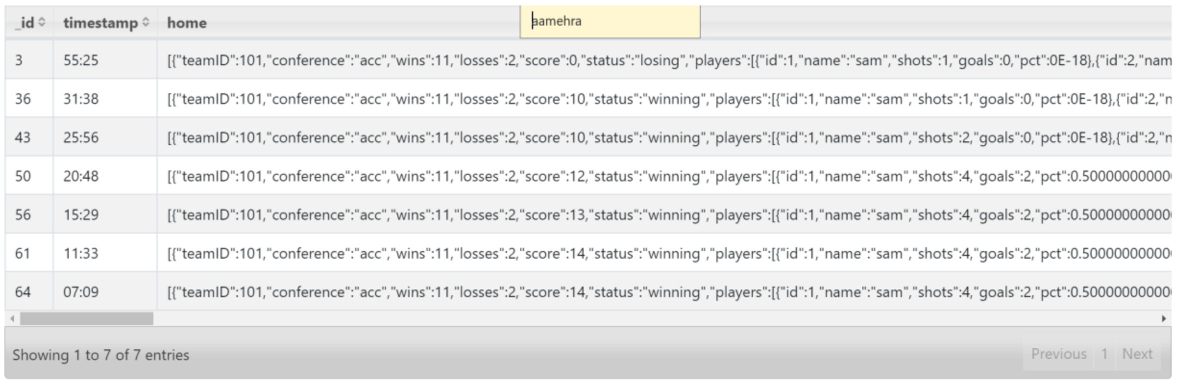
```
+---+--------------------+--------------------+---------+
|_id|                away|                home|timestamp|
+---+--------------------+--------------------+---------+
|  3|[{205, big10, 9, ...|[{acc, 2, [{0, 1,...|    55:25|
| 36|[{205, big10, 9, ...|[{acc, 2, [{0, 1,...|    31:38|
| 43|[{205, big10, 9, ...|[{acc, 2, [{0, 1,...|    25:56|
| 50|[{205, big10, 9, ...|[{acc, 2, [{2, 1,...|    20:48|
| 56|[{205, big10, 9, ...|[{acc, 2, [{2, 1,...|    15:29|
| 61|[{205, big10, 9, ...|[{acc, 2, [{2, 1,...|    11:33|
| 64|[{205, big10, 9, ...|[{acc, 2, [{2, 1,...|    07:09|
+---+--------------------+--------------------+---------+
```

9. Write a drill SQL query to display all the box scores.

select b.`_id`, b.`timestamp`, b.home, b.away
    from mongo.sidearm.boxscores b



```
-- aamehra
select b.`_id`, b.`timestamp`, b.home, b.away
        from mongo.sidearm.boxscores b
```



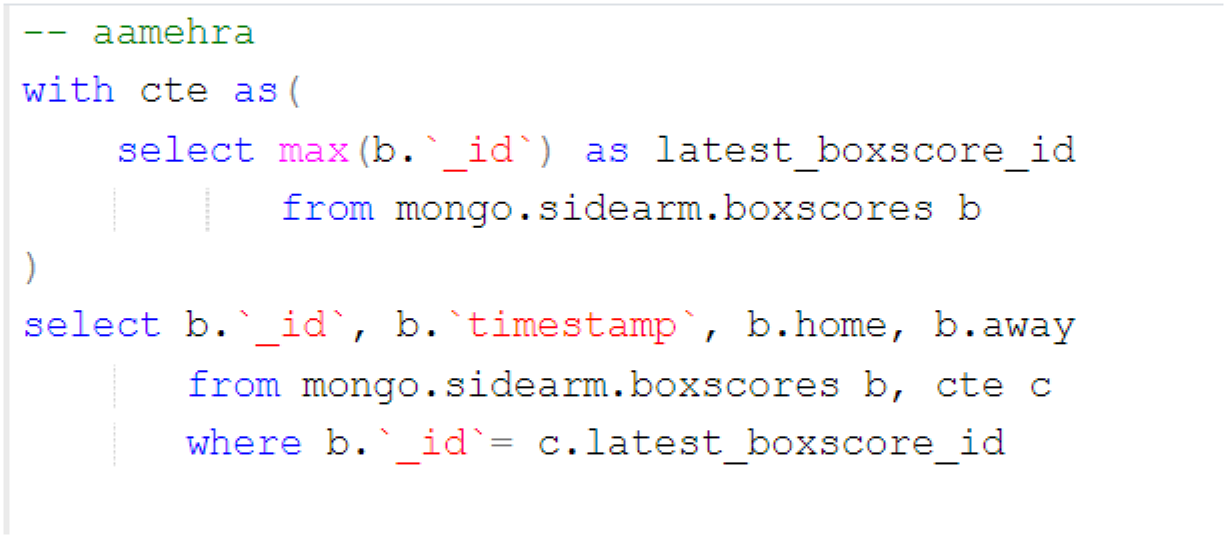
aamehra

_id	timestamp	home
3	55:25	[{"teamID":101,"conference":"acc","wins":11,"losses":2,"score":0,"status":"losing","players":[{"id":1,"name":"sam","shots":1,"goals":0,"pct":0E-18},{"id":2,"nam
36	31:38	[{"teamID":101,"conference":"acc","wins":11,"losses":2,"score":10,"status":"winning","players":[{"id":1,"name":"sam","shots":1,"goals":0,"pct":0E-18},{"id":2,"n
43	25:56	[{"teamID":101,"conference":"acc","wins":11,"losses":2,"score":10,"status":"winning","players":[{"id":1,"name":"sam","shots":2,"goals":0,"pct":0E-18},{"id":2,"n
50	20:48	[{"teamID":101,"conference":"acc","wins":11,"losses":2,"score":12,"status":"winning","players":[{"id":1,"name":"sam","shots":4,"goals":2,"pct":0.500000000000
56	15:29	[{"teamID":101,"conference":"acc","wins":11,"losses":2,"score":13,"status":"winning","players":[{"id":1,"name":"sam","shots":4,"goals":2,"pct":0.500000000000
61	11:33	[{"teamID":101,"conference":"acc","wins":11,"losses":2,"score":14,"status":"winning","players":[{"id":1,"name":"sam","shots":4,"goals":2,"pct":0.500000000000
64	07:09	[{"teamID":101,"conference":"acc","wins":11,"losses":2,"score":14,"status":"winning","players":[{"id":1,"name":"sam","shots":4,"goals":2,"pct":0.500000000000

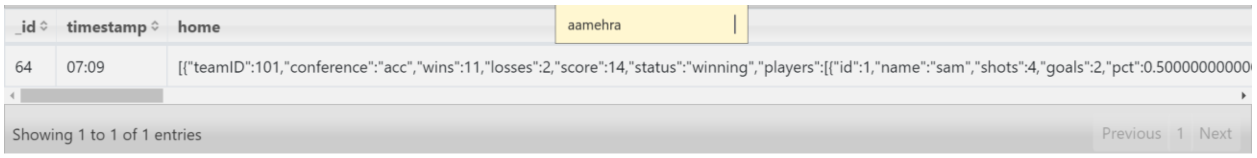
Showing 1 to 7 of 7 entries Previous 1 Next

10. Write a drill SQL query to display the latest box score.

```
with cte as(
   select max(b.`_id`) as latest_boxscore_id
        from mongo.sidearm.boxscores b
)
select b.`_id`, b.`timestamp`, b.home, b.away
     from mongo.sidearm.boxscores b, cte c
     where b.`_id`= c.latest_boxscore_id
```



```
-- aamehra
with cte as(
    select max(b.`_id`) as latest_boxscore_id
            from mongo.sidearm.boxscores b
)
select b.`_id`, b.`timestamp`, b.home, b.away
       from mongo.sidearm.boxscores b, cte c
       where b.`_id`= c.latest_boxscore_id
```



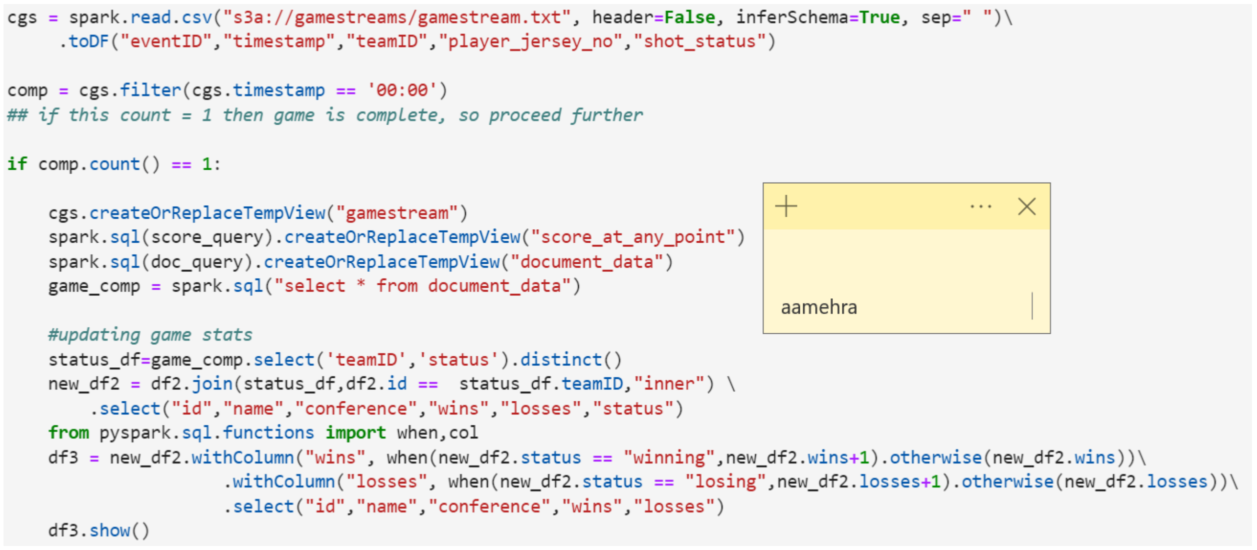
aamehra

_id	timestamp	home
64	07:09	[{"teamID":101,"conference":"acc","wins":11,"losses":2,"score":14,"status":"winning","players":[{"id":1,"name":"sam","shots":4,"goals":2,"pct":0.500000000000

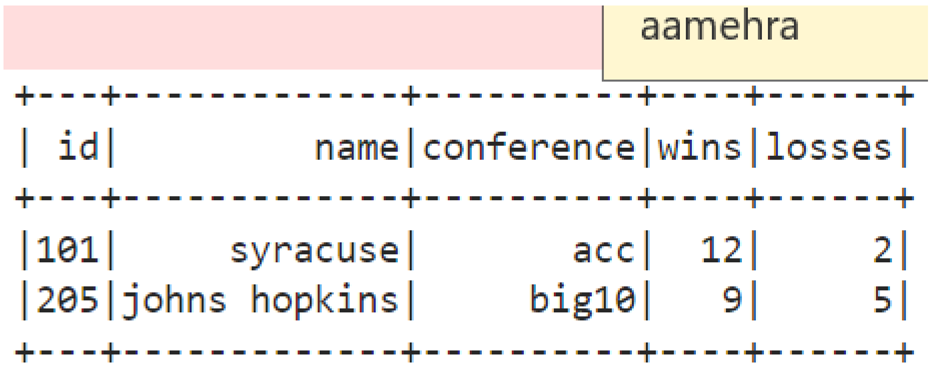
Showing 1 to 1 of 1 entries Previous 1 Next

11. When the game is complete, write pyspark code (in SQL or DataFrame API) update the `wins` and `losses` for the teams in the `teams` table. Specifically, load the `teams` table and update it, then display the updated data frame.

```
cgs = spark.read.csv("s3a://gamestreams/gamestream.txt", header=False,
inferSchema=True, sep=" ")\
    .toDF("eventID","timestamp","teamID","player_jersey_no","shot_status")

comp = cgs.filter(cgs.timestamp == '00:00')
## if this count = 1 then game is complete, so proceed further

if comp.count() == 1:

   cgs.createOrReplaceTempView("gamestream")
   spark.sql(score_query).createOrReplaceTempView("score_at_any_point")
   spark.sql(doc_query).createOrReplaceTempView("document_data")
   game_comp = spark.sql("select * from document_data")

   #updating game stats
   status_df=game_comp.select('teamID','status').distinct()
   new_df2 = df2.join(status_df,df2.id ==  status_df.teamID,"inner") \
      .select("id","name","conference","wins","losses","status")
   from pyspark.sql.functions import when,col
   df3 = new_df2.withColumn("wins", when(new_df2.status ==
"winning",new_df2.wins+1).otherwise(new_df2.wins))\
                .withColumn("losses", when(new_df2.status ==
"losing",new_df2.losses+1).otherwise(new_df2.losses))\
                .select("id","name","conference","wins","losses")
   df3.show()
```



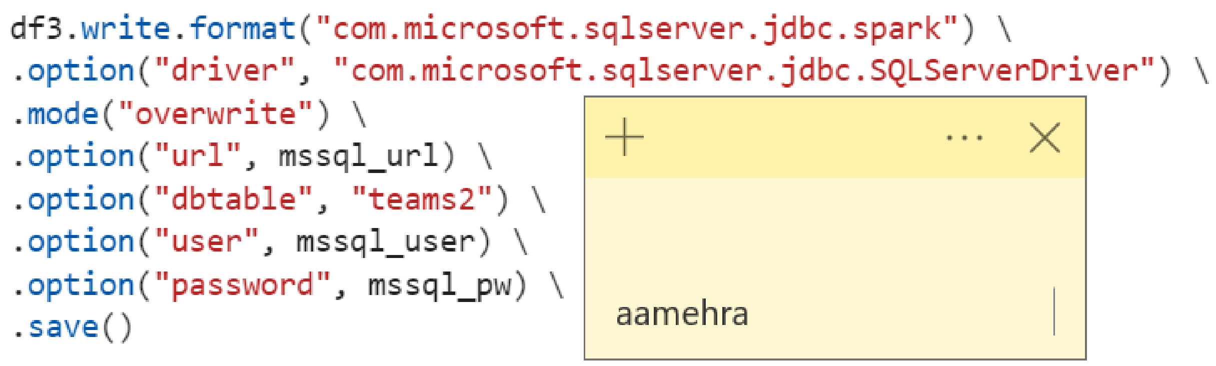
```
cgs = spark.read.csv("s3a://gamestreams/gamestream.txt", header=False, inferSchema=True, sep=" ")\
     .toDF("eventID","timestamp","teamID","player_jersey_no","shot_status")

comp = cgs.filter(cgs.timestamp == '00:00')
## if this count = 1 then game is complete, so proceed further

if comp.count() == 1:

    cgs.createOrReplaceTempView("gamestream")
    spark.sql(score_query).createOrReplaceTempView("score_at_any_point")
    spark.sql(doc_query).createOrReplaceTempView("document_data")
    game_comp = spark.sql("select * from document_data")

    #updating game stats
    status_df=game_comp.select('teamID','status').distinct()
    new_df2 = df2.join(status_df,df2.id ==  status_df.teamID,"inner") \
        .select("id","name","conference","wins","losses","status")
    from pyspark.sql.functions import when,col
    df3 = new_df2.withColumn("wins", when(new_df2.status == "winning",new_df2.wins+1).otherwise(new_df2.wins))\
                     .withColumn("losses", when(new_df2.status == "losing",new_df2.losses+1).otherwise(new_df2.losses))\
                     .select("id","name","conference","wins","losses")
    df3.show()
```



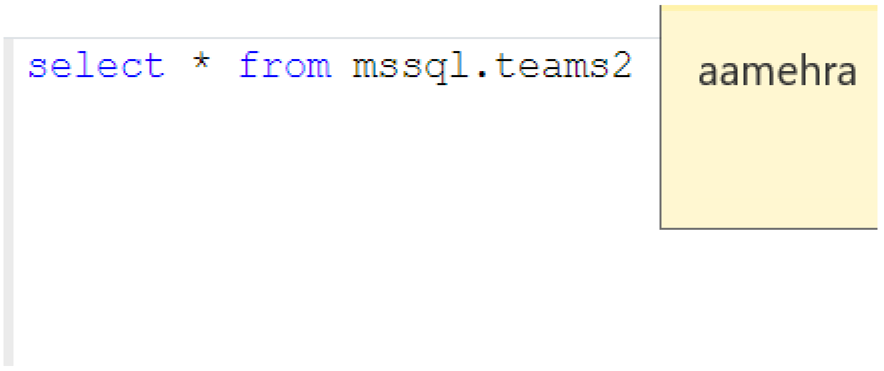
```
+---+-------------+----------+----+------+
| id|         name|conference|wins|losses|
+---+-------------+----------+----+------+
|101|     syracuse|       acc|  12|     2|
|205|johns hopkins|     big10|   9|     5|
+---+-------------+----------+----+------+
```

12. Write pyspark code (in SQL or DataFrame API) to write the updated in question 11 to a new `mssql.sidearmdb.teams2` table.

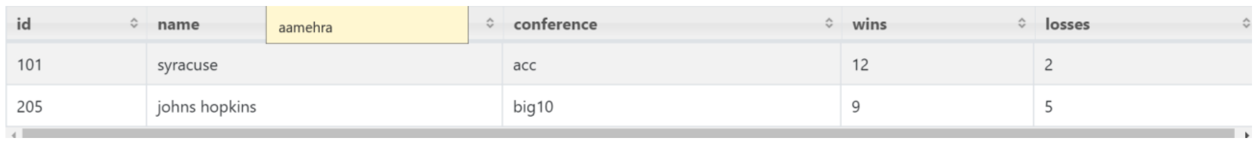
```
df3.write.format("com.microsoft.sqlserver.jdbc.spark") \
    .option("driver", "com.microsoft.sqlserver.jdbc.SQLServerDriver") \
    .mode("overwrite") \
    .option("url", mssql_url) \
    .option("dbtable", "teams2") \
    .option("user", mssql_user) \
    .option("password", mssql_pw) \
    .save()
```



```
df3.write.format("com.microsoft.sqlserver.jdbc.spark") \
.option("driver", "com.microsoft.sqlserver.jdbc.SQLServerDriver") \
.mode("overwrite") \
.option("url", mssql_url) \
.option("dbtable", "teams2") \
.option("user", mssql_user) \
.option("password", mssql_pw) \
.save()
```



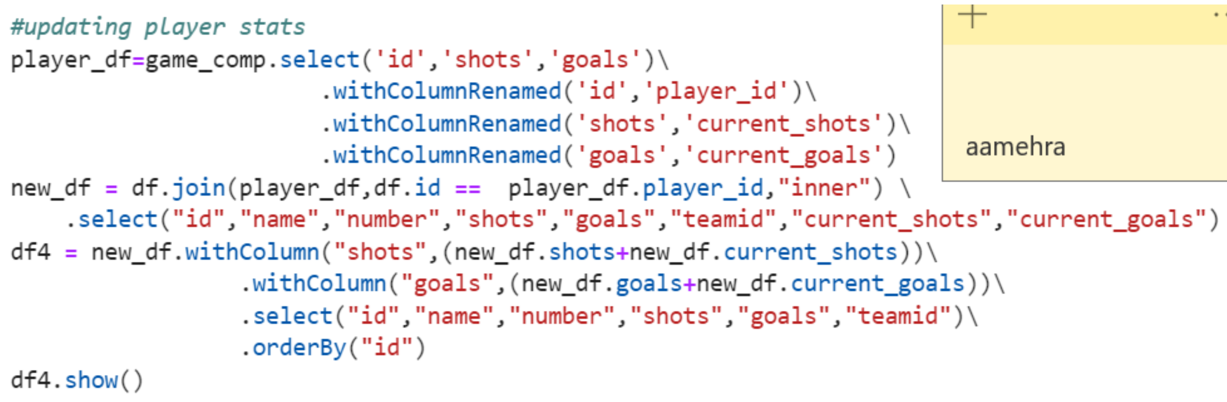
```
select * from mssql.teams2
```



id	name	conference	wins	losses
101	syracuse	acc	12	2
205	johns hopkins	big10	9	5

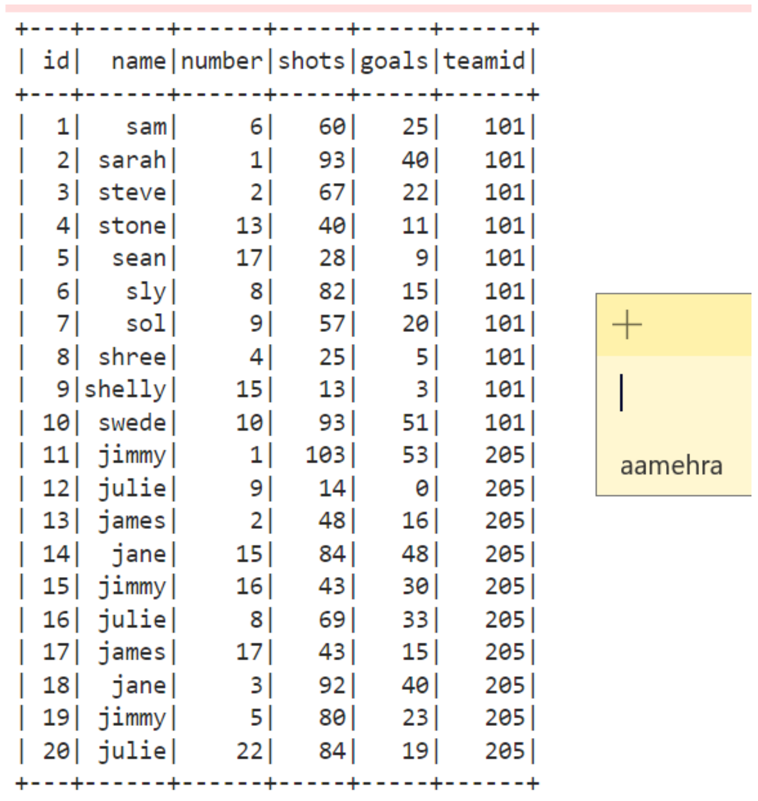
13. When the game is complete, write pyspark code (in SQL or DataFrame API) update the `shots` and `goals` for the players in the `players` table. Specifically, load the `players` table and update it, then display the updated data frame.

```
player_df=game_comp.select('id','shots','goals')\
                    .withColumnRenamed('id','player_id')\
                    .withColumnRenamed('shots','current_shots')\
                    .withColumnRenamed('goals','current_goals')
   new_df = df.join(player_df,df.id ==  player_df.player_id,"inner") \

.select("id","name","number","shots","goals","teamid","current_shots","current_go
als")
   df4 = new_df.withColumn("shots",(new_df.shots+new_df.current_shots))\
                .withColumn("goals",(new_df.goals+new_df.current_goals))\
                .select("id","name","number","shots","goals","teamid")\
                .orderBy("id")
   df4.show()
```



```
#updating player stats
player_df=game_comp.select('id','shots','goals')\
                        .withColumnRenamed('id','player_id')\
                        .withColumnRenamed('shots','current_shots')\
                        .withColumnRenamed('goals','current_goals')
new_df = df.join(player_df,df.id ==  player_df.player_id,"inner") \
    .select("id","name","number","shots","goals","teamid","current_shots","current_goals")
df4 = new_df.withColumn("shots",(new_df.shots+new_df.current_shots))\
                  .withColumn("goals",(new_df.goals+new_df.current_goals))\
                  .select("id","name","number","shots","goals","teamid")\
                  .orderBy("id")
df4.show()
```

aamehra



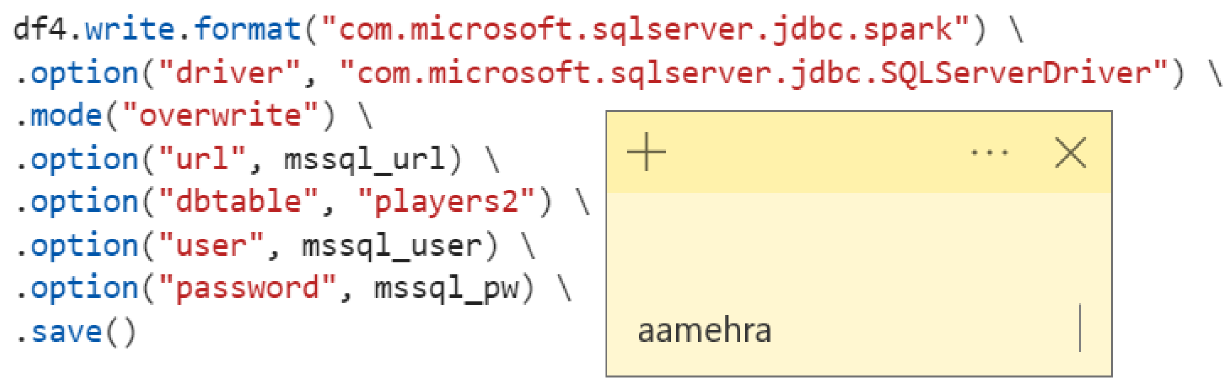
```
+---+------+------+-----+-----+------+
| id|  name|number|shots|goals|teamid|
+---+------+------+-----+-----+------+
|  1|   sam|     6|   60|   25|   101|
|  2| sarah|     1|   93|   40|   101|
|  3| steve|     2|   67|   22|   101|
|  4| stone|    13|   40|   11|   101|
|  5|  sean|    17|   28|    9|   101|
|  6|   sly|     8|   82|   15|   101|
|  7|   sol|     9|   57|   20|   101|
|  8| shree|     4|   25|    5|   101|
|  9|shelly|    15|   13|    3|   101|
| 10| swede|    10|   93|   51|   101|
| 11| jimmy|     1|  103|   53|   205|
| 12| julie|     9|   14|    0|   205|
| 13| james|     2|   48|   16|   205|
| 14|  jane|    15|   84|   48|   205|
| 15| jimmy|    16|   43|   30|   205|
| 16| julie|     8|   69|   33|   205|
| 17| james|    17|   43|   15|   205|
| 18|  jane|     3|   92|   40|   205|
| 19| jimmy|     5|   80|   23|   205|
| 20| julie|    22|   84|   19|   205|
+---+------+------+-----+-----+------+
```

aamehra

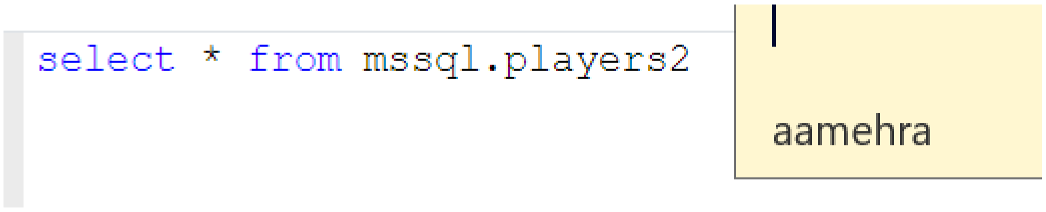
14. Write pyspark code (in SQL or DataFrame API) to write the updated in question 11 to a new `mssql.sidearmdb.players` table.

```
df4.write.format("com.microsoft.sqlserver.jdbc.spark") \
    .option("driver", "com.microsoft.sqlserver.jdbc.SQLServerDriver") \
    .mode("overwrite") \
    .option("url", mssql_url) \
    .option("dbtable", "players2") \
    .option("user", mssql_user) \
    .option("password", mssql_pw) \
```

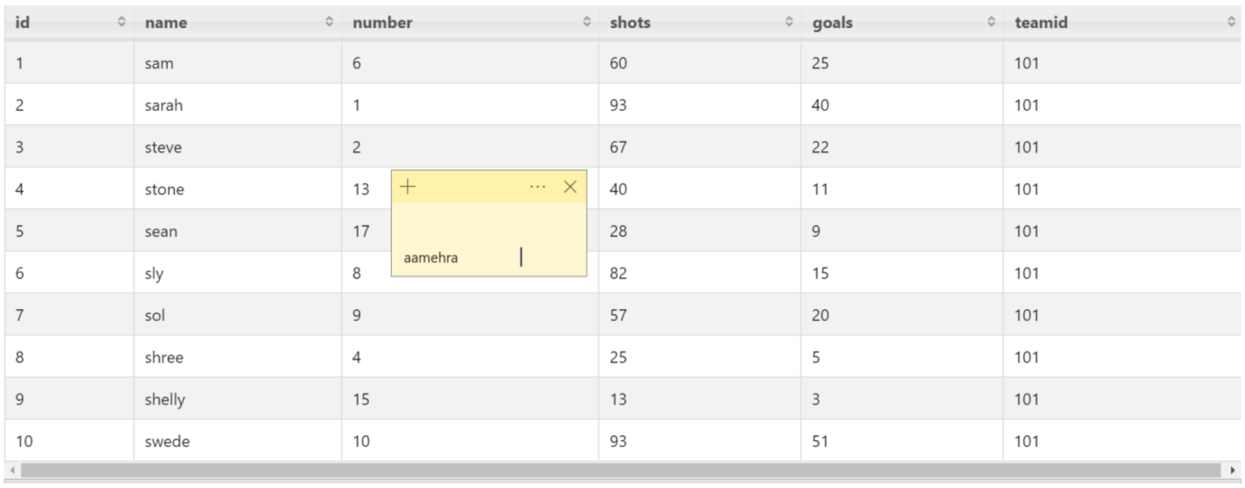
.save()



```
df4.write.format("com.microsoft.sqlserver.jdbc.spark") \
.option("driver", "com.microsoft.sqlserver.jdbc.SQLServerDriver") \
.mode("overwrite") \
.option("url", mssql_url) \
.option("dbtable", "players2") \
.option("user", mssql_user) \
.option("password", mssql_pw) \
.save()
```

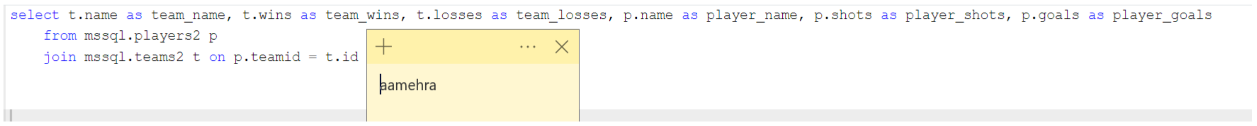


```
select * from mssql.players2
```



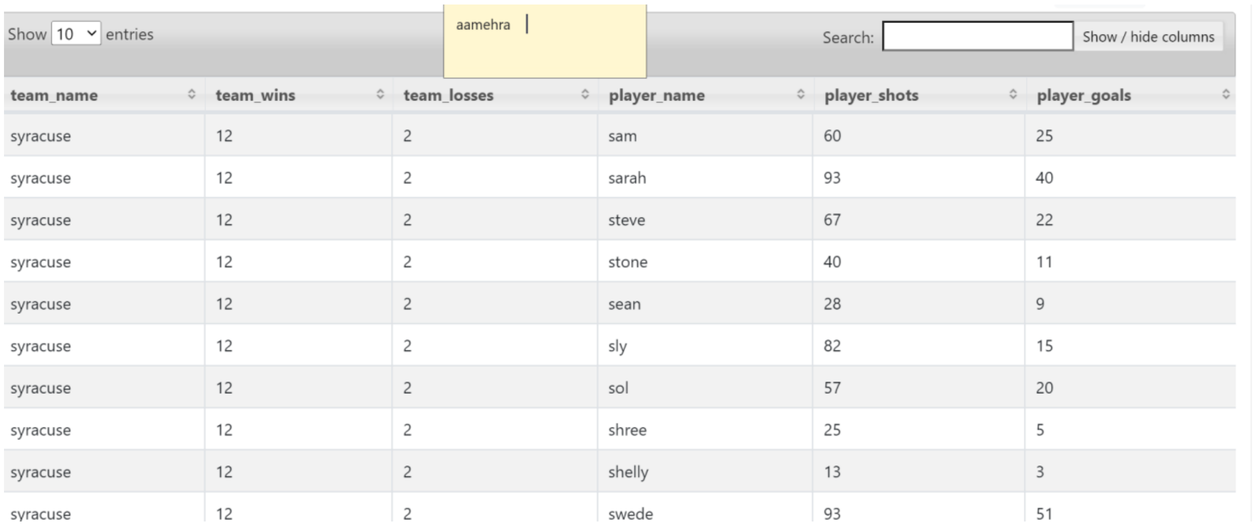
id	name	number	shots	goals	teamid
1	sam	6	60	25	101
2	sarah	1	93	40	101
3	steve	2	67	22	101
4	stone	13	40	11	101
5	sean	17	28	9	101
6	sly	8	82	15	101
7	sol	9	57	20	101
8	shree	4	25	5	101
9	shelly	15	13	3	101
10	swede	10	93	51	101

15. Re-write drill SQL query from question 1 to use the updated `players2` and `teams2` tables.



```
select t.name as team_name, t.wins as team_wins, t.losses as team_losses, p.name as player_name, p.shots as player_shots, p.goals as player_goals
    from mssql.players2 p
    join mssql.teams2 t on p.teamid = t.id
```

aamehra



Show 10 entries

aamehra

Search: Show / hide columns

team_name	team_wins	team_losses	player_name	player_shots	player_goals
syracuse	12	2	sam	60	25
syracuse	12	2	sarah	93	40
syracuse	12	2	steve	67	22
syracuse	12	2	stone	40	11
syracuse	12	2	sean	28	9
syracuse	12	2	sly	82	15
syracuse	12	2	sol	57	20
syracuse	12	2	shree	25	5
syracuse	12	2	shelly	13	3
syracuse	12	2	swede	93	51