In [ ]:
```
import os
import sqlite3
import pandas as pd
import timeit
con = sqlite3.connect('database.sqlite')
cursorObj = con.cursor()
cursorObj1 = con.cursor()
```

In [ ]:
```
from pyspark.sql import SparkSession
spark = SparkSession.builder \
    .master("local[*]") \
    .appName("SparkSQLTest") \
    .getOrCreate()
```

In [ ]:
```
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)
```

In [ ]:
```
cpdf = pd.read_sql_query("select * from Country;", con)
countrydf = spark.createDataFrame(cpdf)

lpdf = pd.read_sql_query("select * from League;", con)
leaguedf = spark.createDataFrame(lpdf)

mpdf = pd.read_sql_query("select * from Match;", con)
matchdf = spark.createDataFrame(mpdf)

ppdf = pd.read_sql_query("select * from Player;", con)
playerdf = spark.createDataFrame(ppdf)

papdf = pd.read_sql_query("select * from Player_Attributes;", con)
playerattridf = spark.createDataFrame(papdf)

tpdf = pd.read_sql_query("select * from Team;", con)
teamdf = spark.createDataFrame(tpdf)

tapdf = pd.read_sql_query("select * from Team_Attributes;", con)
teamattridf = spark.createDataFrame(tapdf)
```

In [ ]:
```
countrydf.createOrReplaceTempView("country")
leaguedf.createOrReplaceTempView("league")
matchdf.createOrReplaceTempView("match")
playerdf.createOrReplaceTempView("player")
playerattridf.createOrReplaceTempView("playerattri")
teamdf.createOrReplaceTempView("team")
teamattridf.createOrReplaceTempView("teamattri")
```

Question 1 (20 points): Write a SQL query that lists all the players born between 1987 and 1990 inclusive, sort them from the oldest to the youngest. The output of this query should be of the form:

```
Player Name | Birthday
```

In [ ]:
```
start = timeit.timeit()
cursorObj.execute("SELECT player_name, birthday "
                  "FROM Player "
                  "WHERE birthday BETWEEN '1987-01-01' AND '1990-12-31' "
                  "ORDER BY birthday"
                  ";")

rows = cursorObj.fetchall()

result = pd.DataFrame(rows, columns=["Player Name", "Birthday"])

end = timeit.timeit()
sqlTimeResult1 = end - start
result
```

In [ ]:
```
# Spark SQL of Question 1
start = timeit.timeit()
result1 = spark.sql("SELECT player_name, birthday "
                  "FROM Player "
                  "WHERE birthday BETWEEN '1987-01-01' AND '1990-12-31' "
                  "ORDER BY birthday"
                  ).show()

end = timeit.timeit()
print("SQL Time = ", sqlTimeResult1)
print("Spark SQL Time = ", end - start)
```

Question 2 (20 points)•: Write a SQL query that ranks all countries and leagues based on the total amount of total goals scored per game in the whole dataset. Sort them by the largest to the smallest amount of goals. Note: Read this carefully. The output of this query should be of the form:

```
Country | League Name |Total Goals Scored#
```

In [ ]:
```
cursorObj.execute("SELECT "
                      "c.name, l.name, SUM(m.home_team_goal + m.away_team_g
                  "FROM "
                      "Match m, "
                      "Country c, "
                      "League l "
                  "WHERE "
                      "c.id = l.country_id and "
                      "c.id = m.country_id and "
                      "l.id = m.league_id "
                  "GROUP BY "
                      "c.id, l.id "
                  "ORDER BY total DESC"
                  ";")

rows = cursorObj.fetchall()

result = pd.DataFrame(rows, columns=["Country ", "League Name ", "Total Goa
result
```

Question 3 (20 points)•: Write a SQL query that ranks all teams by the average of all their attributes (not the players' attributes), sort them from best to worst. The output of this query should be of the form:

```
Team Long Name | Average of Attributes
```

In [ ]:
```
 cursorObj.execute("SELECT t.team_long_name, "
                   "AVG((ta.buildUpPlaySpeed + ta.buildUpPlayDribbling + ta
                   ta.chanceCreationPassing + ta.chanceCreationCrossing + t
                   ta.defencePressure + ta.defenceAggression + ta.defenceTe
                  "FROM Team_Attributes ta, team t "
                  "WHERE t.team_api_id = ta.team_api_id "
                  "GROUP BY ta.team_api_id "
                  "ORDER BY average DESC;")
rows = cursorObj.fetchall()

result = pd.DataFrame(rows, columns=["Team Long Name", "Average of Attribut
result
```

Question 4 (20 points)•: Write a SQL query that ranks all teams by the average of their players' attributes, sort them by descending order displaying only the top 5. The output of this query should be of the form:

```
Team Name| Number of Players | Player Attribute Average
```

```
I could not find a relation between Team and Player, Player Attributes
table to identify Players of each team. So I am considering team
attributes which will provide overall team player's performance average,
and leaving Number of Players column BLANK.
```

In [ ]:
```
 cursorObj.execute("SELECT t.team_long_name, '' as Number, "
                   "AVG((ta.buildUpPlaySpeed + ta.buildUpPlayDribbling + ta
                  "FROM Team_Attributes ta, team t "
                  "WHERE t.team_api_id = ta.team_api_id "
                  "GROUP BY ta.team_api_id "
                  "ORDER BY average DESC LIMIT 5;")
rows = cursorObj.fetchall()


result = pd.DataFrame(rows, columns=["Team Long Name | ", "Number of Player
result
```

Question 5 (40 points)•: Write a SINGLE SQL query that finds the date that had the most goals scored on, per each different season and league. The output of this query should be of the form:

Date (dd/mm/yy) | Season | League Name | Goals scored

In [ ]:
```
cursorObj.execute("SELECT Date(date), season, leagueName, MAX(totalGoals) a
                  " FROM("
                      "SELECT m.date as date, m.season as season, l.id as l
                      "SUM(home_team_goal + away_team_goal) as totalGoals "
                      "FROM Match m, League l "
                      "WHERE m.league_id = l.id "
                      "GROUP BY m.date, m.league_id, m.season"
                    ") "
                  "GROUP BY leagueId, season "
                  "ORDER BY date, season, leagueName"
                  ";")

rows = cursorObj.fetchall()

result = pd.DataFrame(rows, columns=["Date (dd/mm/yy) ", "Season ", "League
result
```

Graduate Student Task (40 points)•: Write a SINGLE SQL query that finds the top 5 teams in terms of goals scored PER league for the 2008/2009 season. The output of this query should be of the form:

Season | League | Rank | Team Name | Goals Scored

In [ ]:
```
cursorObj.execute("SELECT rs.season, rs.lname, Rank, rs.teamname, rs.total
    "SELECT season, lname, teamname, SUM(totalgoals) as total, Rank() over
        "FROM("
                  "SELECT m.season as season, l.name as lname, t.team_long_
                      "FROM Match m, League l, Team t "
                      "WHERE m.league_id = l.id "
                      "and t.team_api_id = m.home_team_api_id "
                      "and m.season='2008/2009' "
                      "GROUP BY m.league_id, t.team_api_id "
                 "UNION "
                  "SELECT m.season as season, l.name as lname, t.team_long_
                      "FROM Match m, League l, Team t "
                      "WHERE m.league_id = l.id "
                      "and t.team_api_id = m.away_team_api_id "
                      "and m.season='2008/2009' "
                      "GROUP BY m.league_id, t.team_api_id "
        ") "
        " GROUP BY lname, teamName "
        ") rs WHERE Rank <=5 "
        "ORDER BY rs.lname, rs.total DESC")

rows = cursorObj.fetchall()

result = pd.DataFrame(rows, columns=["Season ", "League", "Rank", "Team Nam
result
```