```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=

Enter your authorization code:
··········
Mounted at /content/drive
```

```
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
!wget -q https://www-us.apache.org/dist/spark/spark-2.4.4/spark-2.4.4-bin-hadoop2.7.tgz
!tar xf spark-2.4.4-bin-hadoop2.7.tgz
!pip install -q findspark
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-2.4.4-bin-hadoop2.7"
import findspark
findspark.init()
```

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

sc = spark.sparkContext
lines = sc.textFile("spark-2.4.4-bin-hadoop2.7/README.md")
```

```
import os
import sqlite3
```

```
from pyspark.sql import HiveContext, Row
# Or if you can't include the hive requirements
from pyspark.sql import SQLContext, Row

from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)
```

```
import pandas as pd
cnx = sqlite3.connect(r'/content/drive/My Drive/database.sqlite')
```

```
country = pd.read_sql_query("SELECT * FROM Country", cnx)
league = pd.read_sql_query("SELECT * FROM League", cnx)
match = pd.read_sql_query("SELECT * FROM Match", cnx)
player = pd.read_sql_query("SELECT * FROM Player", cnx)
player_attributes = pd.read_sql_query("SELECT * FROM Player_attributes", cnx)
team = pd.read_sql_query("SELECT * FROM Team", cnx)
team_attributes = pd.read_sql_query("SELECT * FROM Team_attributes", cnx)
```

```
sdf_country = spark.createDataFrame(country)
sdf_league = spark.createDataFrame(league)
sdf_match = spark.createDataFrame(match)
```

```
sdf_player = spark.createDataFrame(player)
sdf_player_attributes = spark.createDataFrame(player)
sdf_team = spark.createDataFrame(team)
sdf_team_attributes = spark.createDataFrame(team_attributes)
```

```
sdf_player.createOrReplaceTempView("player")

sdf_country.createOrReplaceTempView("country")

sdf_league.createOrReplaceTempView("league")
sdf_match.createOrReplaceTempView("match")

sdf_player_attributes.createOrReplaceTempView("Player_attributes")
sdf_team.createOrReplaceTempView("team")
sdf_team_attributes.createOrReplaceTempView("team_attributes")
```

```
sdf_player.printSchema()

sdf_country.printSchema()
```

```
root
 |-- id: long (nullable = true)
 |-- player_api_id: long (nullable = true)
 |-- player_name: string (nullable = true)
 |-- player_fifa_api_id: long (nullable = true)
 |-- birthday: string (nullable = true)
 |-- height: double (nullable = true)
 |-- weight: long (nullable = true)

root
 |-- id: long (nullable = true)
 |-- name: string (nullable = true)
```

Answer to Question1 -

```
sqlDF = spark.sql("SELECT player_name, birthday FROM player WHERE birthday between '1987' an
sqlDF.show()
```

```
+--------------------+--------------------+
|         player_name|           birthday|
+--------------------+--------------------+
|       Abdullah Omar|1987-01-01 00:00:00|
|         Ben Alnwick|1987-01-01 00:00:00|
|           Loic Remy|1987-01-02 00:00:00|
|       Robert Milsom|1987-01-02 00:00:00|
|           Celestino|1987-01-02 00:00:00|
|           Luis Dias|1987-01-03 00:00:00|
|        Chris Turner|1987-01-03 00:00:00|
|        Dani Estrada|1987-01-03 00:00:00|
|    Przemyslaw Tyton|1987-01-04 00:00:00|
|           Kay Voser|1987-01-04 00:00:00|
|       Hans Martinez|1987-01-04 00:00:00|
|       Danny Simpson|1987-01-04 00:00:00|
|        Migjen Basha|1987-01-05 00:00:00|
|   Esteban Casagolda|1987-01-05 00:00:00|
|Claudio Lustenberger|1987-01-06 00:00:00|
|       Davide Astori|1987-01-07 00:00:00|
|  Michael McGlinchey|1987-01-07 00:00:00|
|      Stefan Babovic|1987-01-07 00:00:00|
|         Lucas Leiva|1987-01-09 00:00:00|
|     Michele Rinaldi|1987-01-09 00:00:00|
+--------------------+--------------------+
only showing top 20 rows
```

Answer to Question 2 -

```
sqlDF = spark.sql("SELECT country.name, league.name, sum(match.home_team_goal+match.away_tea
sqlDF.show()
```

```
:     +- LogicalRDD [id#4L, country_id#5L, name#6], false
+- SubqueryAlias `match`
   +- LogicalRDD [id#10L, country_id#11L, league_id#12L, season#13, stag

        at org.apache.spark.sql.catalyst.analysis.CheckAnalysis$class.failAnalysi
        at org.apache.spark.sql.catalyst.analysis.Analyzer.failAnalysis(Analyzer.
        at org.apache.spark.sql.catalyst.analysis.CheckAnalysis$$anonfun$checkAna
        at org.apache.spark.sql.catalyst.analysis.CheckAnalysis$$anonfun$checkAna
        at org.apache.spark.sql.catalyst.analysis.CheckAnalysis$$anonfun$checkAna
        at scala.collection.immutable.List.foreach(List.scala:392)
        at org.apache.spark.sql.catalyst.analysis.CheckAnalysis$$anonfun$checkAna
        at org.apache.spark.sql.catalyst.analysis.CheckAnalysis$$anonfun$checkAna
        at org.apache.spark.sql.catalyst.trees.TreeNode.foreachUp(TreeNode.scala:
        at org.apache.spark.sql.catalyst.trees.TreeNode$$anonfun$foreachUp$1.appl
        at org.apache.spark.sql.catalyst.trees.TreeNode$$anonfun$foreachUp$1.appl
        at scala.collection.immutable.List.foreach(List.scala:392)
        at org.apache.spark.sql.catalyst.trees.TreeNode.foreachUp(TreeNode.scala:
        at org.apache.spark.sql.catalyst.trees.TreeNode$$anonfun$foreachUp$1.appl
        at org.apache.spark.sql.catalyst.trees.TreeNode$$anonfun$foreachUp$1.appl
        at scala.collection.immutable.List.foreach(List.scala:392)
        at org.apache.spark.sql.catalyst.trees.TreeNode.foreachUp(TreeNode.scala:
        at org.apache.spark.sql.catalyst.analysis.CheckAnalysis$class.checkAnalys
        at org.apache.spark.sql.catalyst.analysis.Analyzer.checkAnalysis(Analyzer
        at org.apache.spark.sql.catalyst.analysis.Analyzer$$anonfun$executeAndChe
        at org.apache.spark.sql.catalyst.analysis.Analyzer$$anonfun$executeAndChe
        at org.apache.spark.sql.catalyst.plans.logical.AnalysisHelper$.markInAnal
        at org.apache.spark.sql.catalyst.analysis.Analyzer.executeAndCheck(Analyz
        at org.apache.spark.sql.execution.QueryExecution.analyzed$lzycompute(Quer
        at org.apache.spark.sql.execution.QueryExecution.analyzed(QueryExecution.
        at org.apache.spark.sql.execution.QueryExecution.assertAnalyzed(QueryExec
        at org.apache.spark.sql.Dataset$.ofRows(Dataset.scala:78)
        at org.apache.spark.sql.SparkSession.sql(SparkSession.scala:642)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.j
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccess
        at java.lang.reflect.Method.invoke(Method.java:498)
        at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)
        at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)
        at py4j.Gateway.invoke(Gateway.java:282)
        at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)
        at py4j.commands.CallCommand.execute(CallCommand.java:79)
        at py4j.GatewayConnection.run(GatewayConnection.java:238)
        at java.lang.Thread.run(Thread.java:748)


During handling of the above exception, another exception occurred:

AnalysisException                         Traceback (most recent call last)
/content/spark-2.4.4-bin-hadoop2.7/python/pyspark/sql/utils.py in deco(*a, **kw)
     67                                             e.java_exception.getStackTra
     68             if s.startswith('org.apache.spark.sql.AnalysisException: '):
---> 69                 raise AnalysisException(s.split(': ', 1)[1], stackTrace)
     70             if s.startswith('org.apache.spark.sql.catalyst.analysis'):
     71                 raise AnalysisException(s.split(': ', 1)[1], stackTrace)

AnalysisException: "expression 'country.`name`' is neither present in the group b
```

SEARCH STACK OVERFLOW