[1]:
```
from pyspark.sql import SparkSession

from pyspark.sql.functions import *
from pyspark.sql.types import *

spark = SparkSession \
    .builder \
    .appName("Preguntas Obligatorio") \
    .getOrCreate()
```

```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
2024-12-07T20:17:09,017 WARN [Thread-4] org.apache.hadoop.util.NativeCodeLoader - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
```

[2]:
```
athletes = spark.read.csv("/user/ort/obligatorio/refined_tables/athletes", header=True, inferSchema=True)
disciplines = spark.read.csv("/user/ort/obligatorio/refined_tables/disciplines", header=True, inferSchema=True)
events = spark.read.csv("/user/ort/obligatorio/refined_tables/events", header=True, inferSchema=True)
participations = spark.read.csv("/user/ort/obligatorio/refined_tables/participations", header=True, inferSchema=True)
awards = spark.read.csv("/user/ort/obligatorio/refined_tables/awards", header=True, inferSchema=True)

athletes = athletes.withColumn("date_of_birth", to_date(athletes["date_of_birth"], "yyyy-MM-dd"))
events = events.withColumn("date", to_date(events["date"], "yyyy-MM-dd"))
awards = awards.withColumn("award_date", to_date(awards["award_date"], "yyyy-MM-dd"))
```

1. ¿Cuántas medallas de cada tipo (oro, plata, bronce) ha ganado cada país en total?

[3]:
```
medals_by_country = athletes \
    .join(participations, athletes.athlete_id == participations.athlete_id) \
    .join(awards, participations.participation_id == awards.participation_id) \
    .groupBy("country", "medal") \
    .count() \
    .orderBy("country", "medal")

medals_by_country.show()
```

```
+---------+------+-----+
|  country| medal|count|
+---------+------+-----+
|Argentina|  Gold|    5|
|Argentina|Silver|    7|
|  Bolivia|Bronze|    1|
|  Bolivia|  Gold|    2|
|  Bolivia|Silver|    1|
|   Brazil|Bronze|   11|
|   Brazil|  Gold|   17|
|   Brazil|Silver|   15|
|    Chile|Bronze|    2|
| Colombia|Bronze|   10|
| Colombia|  Gold|    3|
| Colombia|Silver|    5|
|  Ecuador|  Gold|    1|
|   Mexico|Bronze|    4|
|   Mexico|  Gold|    4|
|   Mexico|Silver|    1|
| Paraguay|Silver|    1|
|     Peru|Bronze|    5|
|     Peru|  Gold|    2|
|     Peru|Silver|    3|
+---------+------+-----+
```

Persistimos el resultado dentro de la ubicación `/user/ort/obligatorio/analytics/1`

[4]:
```
medals_by_country.coalesce(1).write.csv('/user/ort/obligatorio/analytics/1', header=True, mode='overwrite')
```

2. ¿Qué eventos tienen el promedio de puntajes más alto en cada disciplina?

[5]:
```
participations = participations.withColumnRenamed("event_id", "event_id_2")

# Calcular el promedio de puntajes por disciplina y evento
avg_scores = events \
    .join(participations, events.event_id == participations.event_id_2) \
    .groupBy("discipline_id", "event_id") \
    .avg("score")

# Obtener el puntaje promedio más alto por disciplina
max_scores = avg_scores \
    .groupBy("discipline_id") \
    .max("avg(score)")

# Renombramos columnas por claridad
avg_scores = avg_scores.withColumnRenamed("avg(score)", "avg_score")
max_scores = max_scores.withColumnRenamed("max(avg(score))", "max_avg_score")

# Renombrar columnas para poder hacer select más adelante sin que tire error de ambiguedad
avg_scores = avg_scores.withColumnRenamed("event_id", "event_id_2")
max_scores = max_scores.withColumnRenamed("discipline_id", "discipline_id_2")
max_scores = max_scores.withColumnRenamed("event_id", "event_id_2")
disciplines = disciplines.withColumnRenamed("name", "discipline_name")

# Unir con los eventos y disciplinas para obtener los nombres
result = avg_scores \
    .join(max_scores, (avg_scores.discipline_id == max_scores.discipline_id_2) & (avg_scores.avg_score == max_scores.max_avg_score)) \
    .join(disciplines, avg_scores.discipline_id == disciplines.discipline_id) \
    .join(events, avg_scores.event_id_2 == events.event_id) \
    .select("discipline_name", "event_id", "avg_score") \
    .orderBy("discipline_name")

result.show()
```

```
+---------------+--------+------------------+
|discipline_name|event_id|         avg_score|
+---------------+--------+------------------+
|     basketball|      61|              83.0|
|         boxing|      85|              85.0|
|        cycling|       8|              68.0|
|       football|      92|             72.25|
|           golf|      68|              87.0|
|     gymnastics|      75|              51.0|
|           rugby|      32|              78.5|
|       swimming|      74|88.66666666666667|
|         tennis|      95|              78.0|
|     volleyball|      71|              95.0|
+---------------+--------+------------------+
```

Persistimos el resultado dentro de la ubicación `/user/ort/obligatorio/analytics/2`

[6]:
```
result.coalesce(1).write.csv('/user/ort/obligatorio/analytics/2', header=True, mode='overwrite')
```

3. ¿Cuál es el atleta con mejor puntaje en cada disciplina?

[7]:
```
events = events.withColumnRenamed("discipline_id", "discipline_id_2")
participations = participations.withColumnRenamed("athlete_id", "athlete_id_2")
athletes = athletes.withColumnRenamed("name", "athlete_name")

# Realizar los joins entre todas las tablas involucradas
joined = disciplines \
    .join(events, disciplines.discipline_id == events.discipline_id_2) \
    .join(participations, events.event_id == participations.event_id_2) \
    .join(athletes, participations.athlete_id_2 == athletes.athlete_id) \
    .select("discipline_id", "discipline_name", "athlete_id", "athlete_name", "score")

# Encontrar el puntaje máximo por disciplina
max_scores = joined \
    .groupBy("discipline_id") \
    .max("score")

# Unir con los datos originales para obtener los atletas con el puntaje máximo
best_athletes = joined \
    .join(max_scores, (joined.discipline_id == max_scores.discipline_id) & (joined.score == max_scores['max(score)'])) \
    .select("discipline_name", "athlete_name", "score")

best_athletes.show()
```

```
+---------------+------------+-----+
|discipline_name|athlete_name|score|
+---------------+------------+-----+
|       football|      Eugine|   98|
|     gymnastics|    Faulkner|   98|
|       swimming|       Jayne|   98|
|         tennis|       Roman|  100|
|           golf|    Johnnie|   87|
|     volleyball|       Andie|   95|
|         boxing|        Alix|  100|
|        cycling|       Stacey|   88|
|           rugby|     Heddie|   96|
|     basketball|      Deeann|   99|
+---------------+------------+-----+
```

Persistimos el resultado dentro de la ubicación `/user/ort/obligatorio/analytics/3`

[8]:
```
best_athletes.coalesce(1).write.csv('/user/ort/obligatorio/analytics/3', header=True, mode='overwrite')
```

4. ¿Cuál es el atleta más joven en haber obtenido una medalla de oro?

[9]:
```
# Realizo joins para obtener toda la info de los atletas que obtuvieron medallas de oro
joined = athletes \
    .join(participations, athletes.athlete_id == participations.athlete_id_2) \
    .join(awards, participations.participation_id == awards.participation_id) \
    .filter(col("medal") == "Gold") \
    .withColumn("age_at_award", year(current_date()) - year(col("date_of_birth")))

# Encontrar al atleta más joven
youngest_gold_medalist = joined \
    .groupBy("athlete_id", "athlete_name", "date_of_birth") \
    .min("age_at_award") \
    .orderBy("min(age_at_award)") \
    .limit(1)

# Mostrar los resultados
youngest_gold_medalist.show()
```

```
+----------+------------+-------------+-----------------+
|athlete_id|athlete_name|date_of_birth|min(age_at_award)|
+----------+------------+-------------+-----------------+
|       127|    Marianna|   2008-01-31|               16|
+----------+------------+-------------+-----------------+
```

Persistimos el resultado dentro de la ubicación `/user/ort/obligatorio/analytics/4`

[10]:
```
youngest_gold_medalist.coalesce(1).write.csv('/user/ort/obligatorio/analytics/4', header=True, mode='overwrite')
```

5. ¿Cuál es el puntaje promedio de los atletas para cada país?

[11]:
```
participations_with_country = participations.join(athletes, participations.athlete_id_2 == athletes.athlete_id)

# Agrupar por país, calcular el puntaje promedio y ordenar de mayor a menor
average_scores = participations_with_country \
    .groupBy("country") \
    .agg(round(avg("score"), 2).alias("average_score")) \
    .orderBy("average_score", ascending=False)

average_scores.show()
```

```
+---------+-------------+
|  country|average_score|
+---------+-------------+
|  Bolivia|        56.17|
| Paraguay|         52.0|
|  Ecuador|        50.58|
|     Peru|        50.54|
|   Brazil|        49.67|
| Colombia|        45.35|
|  Uruguay|         44.4|
|    Chile|        44.33|
|Argentina|        43.63|
|   Mexico|        42.08|
+---------+-------------+
```

Persistimos el resultado dentro de la ubicación `/user/ort/obligatorio/analytics/5`

[12]:
```
average_scores.coalesce(1).write.csv('/user/ort/obligatorio/analytics/5', header=True, mode='overwrite')
```