Einsatz und Realisierung von Datenbanken
Spark Hands-On-Session

Aufgaben

Aufgabe 1 (TPC-H in der Spark-Shell)

Führen Sie die Abfragen in der Spark-Shell aus. Als Grundlage für die Abfragen dient das TPC-H Schema.

1. Laden Sie die `region.tbl` Datei als DataFrame Objekt in die Spark-Shell.
2. Ermitteln Sie die Namen aller Regionen.
3. Ermitteln Sie die Zahl der Länder die nicht in Europa liegen.
4. Ermitteln Sie die größte Bestellung aus dem Jahr 1996.
5. Ermitteln Sie welcher europäische Kunde im Jahr 1996 am meisten Geld ausgegeben hat.
6. Ermitteln Sie welche Unternehmen keine Kunden in Europa haben.

Lösungsvorschlag zu Aufgabe 1

1. Um die Datei als DataFrame zu laden, braucht man das Format der Datei, in diesem Fall CSV, das Schema der Daten, das Zeichen, mit dem die Spalten in der CSV-Datei getrennt werden, sowie den Pfad. Für das Schema gibt man für jede Spalte den Namen und den Typ an, sowie ob das Feld auch einen null-Wert enthalten darf:

```
def region: DataFrame = {
  spark.read.format("csv").schema(StructType(
    List(
      StructField("r_regionkey", IntegerType, false),
      StructField("r_name", StringType, false),
      StructField("r_comment", StringType, false)
    )
  )).option("delimiter", "|").load(DATA_PATH + /region.tbl")
}
```

2. Mithilfe der `select` Funktion können bestimmte Spalten ausgewählt werden:

```
region.select($"r_name").show
```

3. Zuerst wird aus dem DataFrame `region` Europa entfernt. Danach wird das gefilterte DataFrame mit `nation` gejoined um alle nicht-europäischen Länder zu finden. Statt der Aktion `show` wird die Aktion `count` verwendet, die keinen Text ausgibt, sondern einen Integer zurückgibt:

```
val notEurope = region.filter($"r_name" =!= "EUROPE")
val nonEuropeanCountries = nation
  .join(notEurope, $"r_regionkey" === $"n_regionkey", "leftsemi")
nonEuropeanCountries.count
```

4. Zuerst werden alle Bestellungen herausgesucht, die im Jahr 1996 getätigt wurden. Danach sucht man alle Einträge aus dem `lineitem` DataFrame heraus, die zu einer der Bestellungen aus dem Jahr 1996 gehören. Um den Gesamtumfang der Bestellung zu ermitteln summiert man die Menge der `lineitems` pro Bestellung auf. Der größte Umfang kann dann durch eine Sortierung ermittelt werden:

```
val ordersOf1996 = orders.filter(year($"o_orderdate") === 1996)
val biggestOrder = lineitem
  .join(ordersOf1996, $"o_orderkey" === $"l_orderkey")
  .groupBy($"o_orderkey")
  .agg(sum($"l_quantity").as("size"))
  .orderBy($"size".desc)
  .limit(1)
biggestOrder.show
```

5. Zuerst werden alle Kunden herausgesucht aus europäischen Ländern mithilfe der `nation` und `region` DataFrames. Die Bestellungen werden dann wie bereits in Aufgabe 4 nach dem Jahr 1996 gefiltert. Danach werden die Gesamtpreise der Bestellungen pro Kunde aufsummiert. Der Kunde mit den höchsten Ausgaben kann dann wieder mit einer Sortierung ermittelt werden:

```
val nationsOfEurope = region
  .filter($"r_name" === "EUROPE")
  .join(nation, $"r_regionkey" === $"n_regionkey")
  .select($"n_nationkey")
val customerOfEurope = customer
  .join(nationsOfEurope, $"n_nationkey" === $"c_nationkey", "leftsemi")
  .select($"c_custkey")
val ordersOf1996 = orders.filter(year($"o_orderdate") === 1996)
val customerWithHighestCosts = ordersOf1996
  .join(customerOfEurope, $"o_custkey" === $"c_custkey")
  .join(lineitem, $"o_orderkey" === $"l_orderkey")
  .groupBy($"c_custkey")
  .agg(sum($"l_quantity" * $"l_extendedprice").as("costs"))
  .orderBy($"costs".desc)
  .limit(1)
customerWithHighestCosts.show
```

6. Um herauszufinden welche Unternehmen keine europäische Kunden haben, müssen alle Einträge in `lineitem` ermittelt werden, die von europäischen Kunden stammen. Dafür werden die DataFrames `region`, `nation`, `customer`, `orders` und `lineitem` miteinander verbunden. Mit einem anti-Join von `supplier` mit der Ergebnis-Relation bleiben nur die Unternehmen übrig, die keine Kunden in Europa haben:

```
val lineitemsOrderedByEuropeans = region
  .filter($"r_name" === "EUROPE")
  .join(nation, $"r_regionkey" === $"n_regionkey")
  .join(customer, $"n_nationkey" === $"c_nationkey")
  .join(orders, $"o_custkey" === $"c_custkey")
  .join(lineitem, $"o_orderkey" === $"l_orderkey")
val companiesWithoutEuropeanCustomers = supplier
  .join(lineitemsOrderedByEuropeans, $"l_suppkey" === $"s_suppkey", "leftanti")
companiesWithoutEuropeanCustomers.show
```