TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

Übung zur Vorlesung *Einsatz und Realisierung von Datenbanken* im SoSe25

Alice Rey, Maximilian Reif, Tobias Goetz (i3erdb@in.tum.de)
http://db.in.tum.de/teaching/ss25/impldb/

Blatt Nr. 13

Hausaufgabe 1

Führen Sie die folgenden Abfragen in der Spark-Shell aus. Als Grundlage für die Abfragen dient das TPC-H Schema. Laden Sie dazu die TPC-H Daten wie in der Vorlesung gezeigt in die Spark-Shell.

(a) Laden Sie die `region.tbl` Datei als DataFrame Objekt in die Spark-Shell.

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:

```
val region = 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("region.tbl")
```

(b) Ermitteln Sie die Namen aller Regionen.

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

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

(c) Ermitteln Sie die Zahl der Länder die nicht in Europa liegen.

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
```

(d) Ermitteln Sie die größte Bestellung aus dem Jahr 1996.

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.

```
val ordersOf1996 = orders
    .filter(year($"o_orderdate") === 1996)

val orderSizes = ordersOf1996
    .join(lineitem, $"o_orderkey" === $"l_orderkey")
    .groupBy($"o_orderkey")
    .agg(sum($"l_quantity").as("size"))

val maxOrderSize = orderSizes
    .agg(max($"size").as("max_size"))

val biggestOrder = orderSizes
    .join(maxOrderSize, $"size" === $"max_size")
    .select($"o_orderkey", $"size")

biggestOrder.show
```

(e) Ermitteln Sie welcher europäische Kunde im Jahr 1996 am meisten Geld ausgegeben hat.

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 europeanCountries = nation
    .join(region, $"r_regionkey" === $"n_regionkey")
    .filter($"r_name" === "EUROPE")
    .select($"n_nationkey")

val customerOfEurope = customer
    .join(europeanCountries, $"n_nationkey" === $"c_nationkey", "leftsemi")
    .select($"c_custkey")

val ordersOf1996 = orders
    .filter(year($"o_orderdate") === 1996)

val customerWithCosts = ordersOf1996
    .join(customerOfEurope, $"o_custkey" === $"c_custkey")
    .groupBy($"c_custkey")
    .agg(sum($"o_totalprice").as("costs"))

val highestCosts = customerWithCosts
    .agg(max($"costs").as("highest_costs"))

val customerWithHighestCosts = customerWithCosts
    .join(highestCosts, $"costs" === $"highest_costs")
    .select($"c_custkey", $"costs")

customerWithHighestCosts.show
```

(f) Ermitteln Sie welche Unternehmen keine Kunden in Europa haben.

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
```

Hausaufgabe 2

Führen Sie die folgenden Abfragen in der Spark-Shell aus. Als Grundlage für die Abfragen dient das TPC-H Schema. Laden Sie dazu die TPC-H Daten wie in der Vorlesung gezeigt in die Spark-Shell.

(a) Ermitteln Sie pro Marktsegment die Anzahl der Bestellungen in 1997.

(b) Ermitteln Sie die Zahl der Kunden und Lieferanten pro Land.

(c) Ermitteln Sie die Stückzahlen der verschiedenen Bauteile in Deutschland.

(d) Ermitteln Sie, welche Kunden kein *goldenrod lavender spring chocolate lace* bestellt haben.

Lösung:

(a) Ermitteln Sie pro Marktsegment die Anzahl der Bestellungen in 1997.

```
val ordersOf1997 = orders.where(year($"o_orderdate") === 1997)
val customerOrders = ordersOf1997.join(customer, $"c_custkey" === $"o_custkey")
val mktSegmentOrders = customerOrders
   .groupBy($"c_mktsegment")
   .agg(count($"o_orderkey").as("orderCount"))

mktSegmentOrders.show
```

(b) Ermitteln Sie die Zahl der Kunden und Lieferanten pro Land.

```
val countryCustomer = customer.groupBy($"c_nationkey")
   .agg(count($"c_custkey").as("customerCount"))
val countrySupplier = supplier.groupBy($"s_nationkey")
   .agg(count($"s_suppkey").as("supplierCount"))
val countryCustSupp = nation
   .join(countryCustomer, $"c_nationkey" === $"n_nationkey")
   .join(countrySupplier, $"s_nationkey" === $"n_nationkey")

countryCustSupp
   .select($"n_nationkey", $"n_name", $"customerCount", $"supplierCount")
   .show
```

(c) Ermitteln Sie die Stückzahlen der verschiedenen Bauteile in Deutschland.

```
val germany = nation.filter($"n_name" === "GERMANY")
val germanSupplier = supplier
   .join(germany, $"n_nationkey" === $"s_nationkey", "leftsemi")
val germanParts = partsupp
   .join(germanSupplier, $"ps_suppkey" === $"s_suppkey","leftsemi")
   .groupBy($"ps_partkey")
   .agg(sum($"ps_availqty").as("stueckzahl"))

germanParts.show
```

(d) Ermitteln Sie, welche Kunden kein *goldenrod lavender spring chocolate lace* bestellt haben.

```
val goldenRodLavenderSpringChoclateLaceOrders = part
   .filter($"p_name" === "goldenrod␣lavender␣spring␣chocolate␣lace")
   .join(partsupp, $"ps_partkey" === $"p_partkey")
   .join(lineitem, $"l_partkey" === $"ps_partkey")
   .join(orders, $"o_orderkey" === $"l_orderkey")
   .select($"o_custkey")
   .distinct

val noChocolateCustomer = customer.join(
   goldenRodLavenderSpringChoclateLaceOrders,
   $"o_custkey" === $"c_custkey",
   "leftanti")

noChocolateCustomer.show
```