

Bericht Leistungsnachweis Modul DB & DWH

Einleitung

In unserem fiktiven Case geht es darum, ein (online zugängliches) automatisiertes Tool zu erstellen, anhand dessen die Entscheidungsträgerinnen unserer Firma vorbereitete Analysen zum Gewinn und den Kunden des Unternehmens vornehmen können. Wir stellen uns vor, dass die Entscheidungsträgerinnen sehr oft unterwegs sind und deshalb einen Onlinezugang zum Tool benötigen - es muss also alles webbasiert sein.

Wir haben uns daher folgende Ziele gesetzt:

- Automatisches Bearbeiten / Reinigen der uns interessierenden operativen Daten (OLTP-erzeugt) und Integrieren dieser Daten in eine MySQL Datenbank.
- Online-Hosting der MySQL Datenbank.
- Erstellen einer Web-Applikation, welche folgende Analysen an der Datenbank vornimmt:
 - Zeitverlauf von des nach Monat aggregierten Gewinns und ein Vorhersage des weiteren Gewinnverlaufs.
 - Auflisten (tabellarisch und graphisch) der Top-fünf Gewinnbringer in den Bereichen a) Bundesstaat, b) Kalendermonat und c) Jobbezeichnung.
 - Visuelle Darstellung von allfälligen Kundengruppen, die ähnliches Kaufverhalten zeigen (für kundenspezifische Kampagnen).

Für die Beantwortung unserer Fragestellung ist eine spezielle Kombination von Datentabellen aus den operativen Daten nötig. Diese Kombination ist in einem "Sternschema" als ERM im PDF "sternschema_erm_leistungsnachweis_dwh" ersichtlich. Auf dieses Schema beziehen sich die Beschreibungen.

Im Folgenden werden die Arbeitsschritte erläutert, die zum Erreichen unserer Ziele nötig waren. Aufgeteilt ist der Bericht in

- A) "Backend" (Beschreibungen zu Pentaho PDI und MySQL Arbeitsschritten) und
- B) "Frontend" ("Shiny" Webapplikation der R-Analysen).

A) Backend (MySQL & Pentaho PDI)

Ziel des Backends ist es, die für das vereinbarte Sternschema notwendigen Datenbestände aus der Northwind Database zu exportieren, zu bereinigen, zu reorganisieren und in eine neue Datenbank hochzuladen.

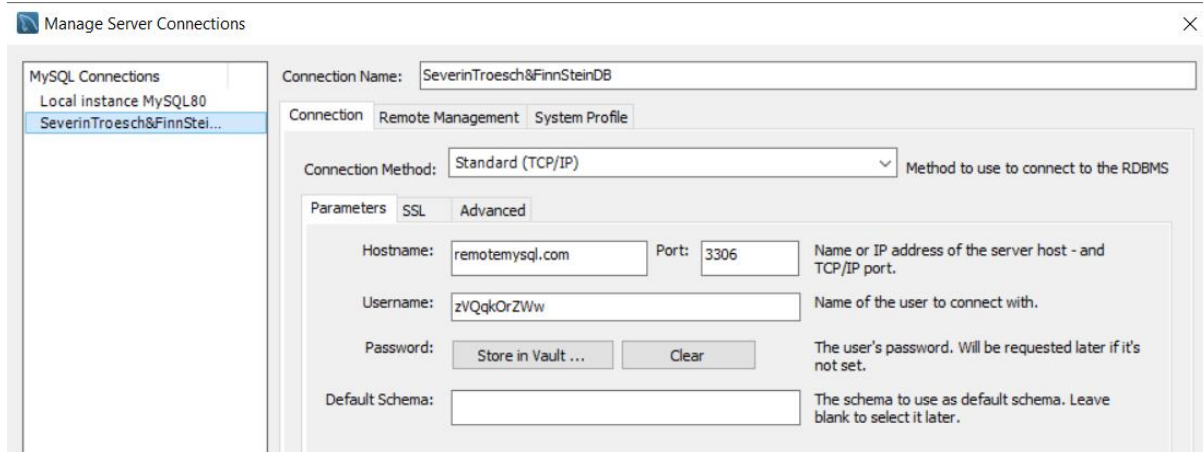
Dabei sind folgende Schritte zu unterscheiden:

- (1) Erstellen einer Ziel-Datenbank mit der Struktur des im ERM vereinbarten Sternschemas
- (2) Export der nötigen Tabellen als CSV aus Northwind Database.
- (3) Import CSV Dateien in Pentaho
- (4) Bereinigung der aus den CSVs importierten Daten (Duplikate in Produkten & neg. Discounts)
- (5) Reorganisation der Daten zu den für das Sternschema nötigen Einzel-Tabellen
- (6) Export der Tabellen in die zuvor erstellte Ziel-Datenbank

A.1 Erstellen der Ziel-Datenbank

Referenzdatei: CreateNorthwindDataCube2.sql

In einem ersten Schritt erstellen wir die Verbindung zwischen MySQL Workbench und der online auf <https://remotemysql.com/> erstellten Datenbank "zVQqkOrZWw".

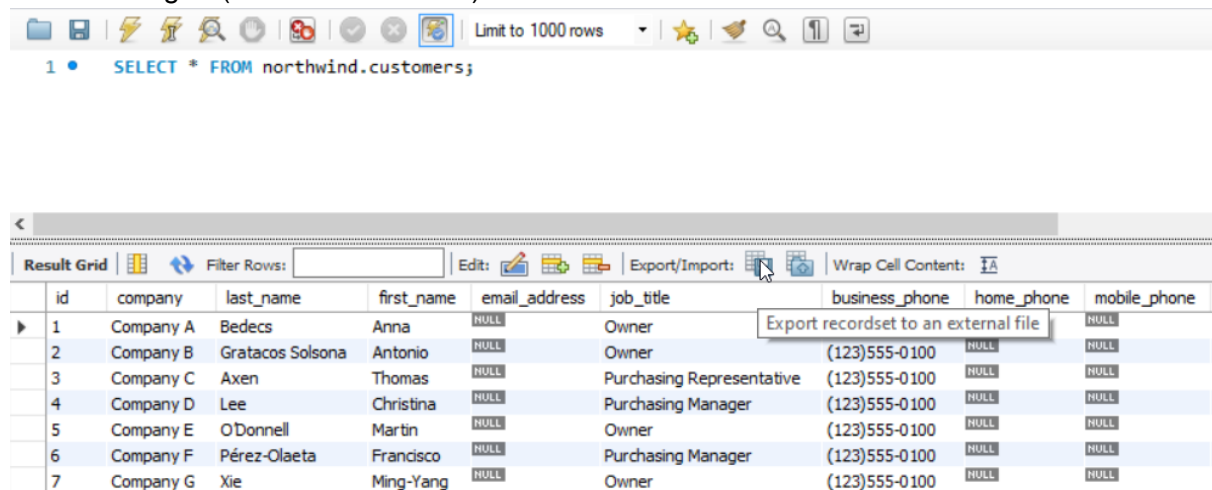


Anschliessend erstellen wir die Struktur der zuvor im ERM vereinbarten Tabellen und Bezüge. Speziell zu beachten sind dabei die von der Tabelle "facts" ausgehenden Fremdschlüsselbezüge zu allen anderen Tabellen des Sternschemas. Der komplette Code ist in eingangs genannter Datei enthalten, die dem Bericht beiliegt.

```
UNIQUE (customer_id, date_id, destination_id, employee_id, product_id),  
CONSTRAINT FK_product FOREIGN KEY(product_id) REFERENCES product(id),  
CONSTRAINT FK_customer FOREIGN KEY(customer_id) REFERENCES customer(id),  
CONSTRAINT FK_date FOREIGN KEY(date_id) REFERENCES date(id),  
CONSTRAINT FK_destination FOREIGN KEY(destination_id) REFERENCES destination(id),  
CONSTRAINT FK_employee FOREIGN KEY(employee_id) REFERENCES employee(id)
```

A.2 Export der nötigen Tabellen als CSV aus der Northwind Database

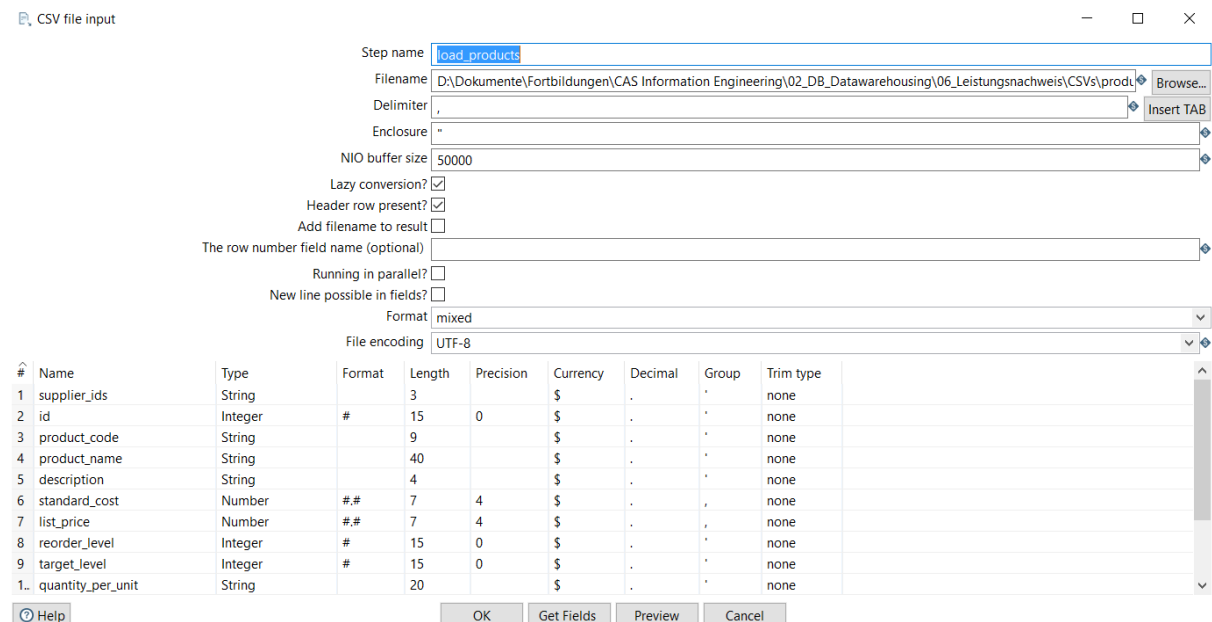
Dieser Schritt lässt sich einfach über das visuelle interface der MySQL Workbench bewerkstelligen (siehe Screenshots).



A.3 Import CSV Dateien in Pentaho

Referenzdatei für alle PentahoOperationen: ETL-SeverinTroesch&FinnStein7.ktr

Wir laden die CSV Dateien über den Step "Input" => "CSV file Input" in die Staging Area von Pentaho. Für jede der Dateien erhalten wir über "Get Fields" die zu importierenden Spalten. Als File encoding wählen wir "UTF-8". Über Preview prüfen wir, ob die importierten Daten korrekt dargestellt werden, bzw. ob Pentaho das korrekte Ausgabensformat wählt.



A.4.1 Bereinigung der Daten (Duplikate in Tabelle products)

Die Tabelle "products" enthält Duplikate, die sich nur beim Eintrag "id" unterscheiden. Wir wollen diese beseitigen und einen kurzen Excel-Bericht mit den zu löschenden Duplikaten für unsere Datenerfassungsabteilung erstellen.

Hierfür sortieren wir in einem ersten Schritt die Tabelle “products” nach “product_code” und anschliessend “id”. Letzteres ist notwendig, da wir immer den Produkteintrag mit der niedrigeren “id” behalten wollen. Step: “Transform” => “Sort Rows”.

Sort rows

Step name:

Sort directory:

TMP-file prefix:

Sort size (rows in memory):

Free memory threshold (in %):

Compress TMP Files? ☐

Only pass unique rows? (verifies keys only) ☐

Fields :

| # | Fieldname | Ascending | Case sensitive compare? | Sort based on current locale? | Collator Strength | P |
|---|--------------|-----------|-------------------------|-------------------------------|-------------------|---|
| 1 | product_code | Y | N | N | 0 | N |
| 2 | id | Y | N | N | 0 | N |

Anschliessend filtern wir über den Step “Transform” => “Unique Rows” die Duplikate der Tabelle heraus. Hierfür ist es wichtig, dass die Reihen der Tabelle zuvor sortiert wurden, da nur direkte aufeinander folgende Duplikate erkannt werden. Als Massstab für ein Duplikat wählen wir alle Spalten ausser “id”, da letztere eindeutig für alle Einträge ist.

Unique rows

Step name:

Settings

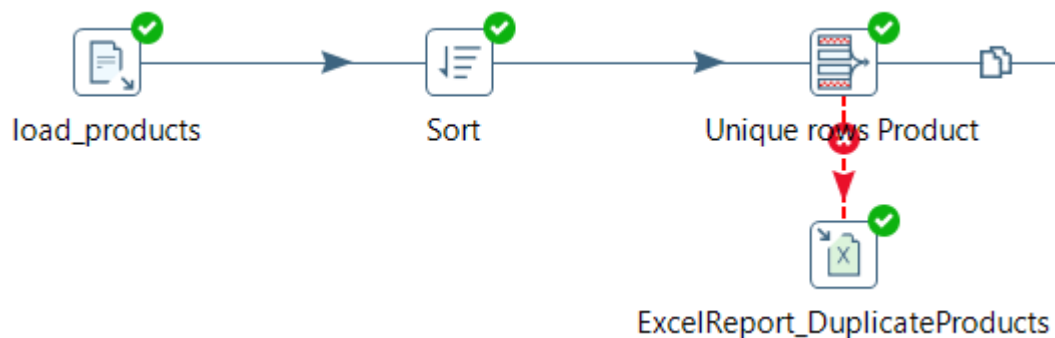
Add counter to output? ☐ Counter field:

Redirect duplicate row ☒ Error description:

Fields to compare on (no entries means: compare complete row)

| # | Fieldname | Ignore case |
|-----|--------------------------|-------------|
| 1 | supplier_ids | N |
| 2 | product_code | N |
| 3 | product_name | N |
| 4 | description | N |
| 5 | standard_cost | N |
| 6 | list_price | N |
| 7 | reorder_level | N |
| 8 | target_level | N |
| 9 | quantity_per_unit | N |
| 1.. | discontinued | N |
| 1.. | minimum_reorder_quantity | N |
| 1.. | category | N |
| 1.. | attachments | N |

Als “Error Handling of Step” exportieren wir einen Excel Report mit den Duplikat-Einträgen, den wir an unsere Datenerfassung weiterreichen können (Step: “Output” => “Microsoft Excel output”).



A.4.2 Bereinigung der Daten (Reformatierung Discounts)

In der Tabelle "order_details" ist die Variable discount als negativer Wert erfasst. In einem ersten Schritt wandeln wir diese in eine positive Variable um. Diese ist wichtig für unser Vorhaben, da wir als finalen fact den "Gewinn_nach_Discount" pro Bestellung ausweisen wollen. Zuerst wandeln wir daher alle negativen Werte in positive um. Anschliessend stellen wir aber fest, dass der Discount pro Bestellung teilweise höher ist als der Listenpreis des entsprechenden Produktes. Daher gehen wir davon aus, dass es sich um eine Angabe in Prozentpunkten handelt und erstellen eine Variable "discountFactor", welche wir mit den Listenpreisen multiplizieren können um den finalen Verkaufspreis zu erhalten.
(Step: "Scripting" => "Formula")

fx Formula

Step name:

Fields:

| # | New field | Formula | Value type | Length | Precision | Replace value |
|---|----------------|---|------------|--------|-----------|---------------|
| 1 | discountFactor | IF([discount]<0;(1-([discount]*-1)/100);1-[discount]/100) | Number | | | |

A.5.1 Reorganisation der Daten als Sternschema (Tabelle facts - Fremdschlüssel)

Als Basis für die Tabelle "facts" verwenden wir die Tabelle "order_details", da hier bereits Fremdschlüssel zu den Tabellen "orders" und "products", sowie wichtige Fakten für die Berechnung unserer Zielgrössen enthalten sind ("discountFactor", "unit_price" und "quantity"). Anschliessend reichern wir die Tabelle über den Step: "Lookup" => "Stream lookup" mit den Produktkosten sowie den Fremdschlüsseln aller verbleibenden Tabellen an.

Stream lookup

Step name:

Lookup step:

The key(s) to look up the value(s):

| # | Field | LookupField |
|---|------------|-------------|
| 1 | product_id | id |

Specify the fields to retrieve :

| # | Field | New name | Default | Type |
|---|---------------|----------|---------|--------|
| 1 | standard_cost | | NULL | Number |


Preserve memory (costs CPU) ☒

Key and value are exactly one integer field ☐

Use sorted list (i.s.o. hashtable) ☒

A.5.2 Reorganisation der Daten als Sternschema (Tabelle facts - Zielfakten)

Über den Step: "Scripting" => "Formula" berechnen wir alle nötigen Zielfakten für die von uns angestrebten Auswertungen. D.h. "Umsatz_vor_Discount", "Umsatz_nach_Discount", "Gewinn_vor_Discount" und "Gewinn_nach_Discount".

 Formula

Step name


Fields:

| # | New field | Formula | Value type |
|---|----------------------|---|------------|
| 1 | Umsatz_vor_Discount | [quantity]*[unit_price] | Number |
| 2 | Umsatz_nach_Discount | [quantity]*[unit_price]*[discountFactor] | Number |
| 3 | Gewinn_vor_Discount | [quantity]*[unit_price]-[quantity]*[standard_cost] | Number |
| 4 | Gewinn_nach_Discount | [quantity]*[unit_price]*[discountFactor]-[quantity]*[standard_cost] | Number |

A.5.3 Reorganisation der Daten als Sternschema (Tabellen date und destination aus Tabelle orders erstellen)

Für das Sternschema benötigen wir eine Tabelle "date" mit den Bestelldaten, sowie Jahr, Monat und Kalenderwoche. Hierfür lösen wir die Spalte "order_date" als separate Tabelle aus der tabelle "orders" heraus. Anschliessend eliminieren wir alle Duplikate über die Steps "Sort rows" und "Unique rows", analog dem Vorgehen bei der Tabelle "products".

Anschliessend erstellen wir über den Step: "Transform" => "Calculator" die zusätzlichen Spalten "year", "month", "cw", "weekday" aus dem Datum heraus. Hierfür gibt es in Pentaho spezifische Funktionen.

 Calculator

Step name

☒ Throw an error on non existing files

Fields:

| # | New field | Calculation | Field A | Field B | Field C | Value type | Length |
|---|-----------|------------------------|------------|---------|---------|------------|--------|
| 1 | Year | Year of date A | order_date | | | None | |
| 2 | Month | Month of date A | order_date | | | None | |
| 3 | CW | Week of year of date A | order_date | | | None | |
| 4 | Weekday | Day of week of date A | order_date | | | None | |

Zuletzt erstellen wir über den Step: "Transform" => "Add sequence" eine neue id Variable für die Tabelle und verknüpfen diese via Lookup Step mit der Facts Tabelle. Das konkrete order_date für den Lookup match hatten wir der Tabelle "facts" bereits zuvor beigefügt.

Add sequence

Step name:

Name of value:

Use a database to generate the sequence

Use DB to get sequence? ☐

Connection: Edit... New... Wizard...

Schema name: Schemas...

Sequence name: Sequences...

Use a transformation counter to generate the sequence

Use counter to calculate sequence? ☒

Counter name (optional):

Start at value:

Increment by:

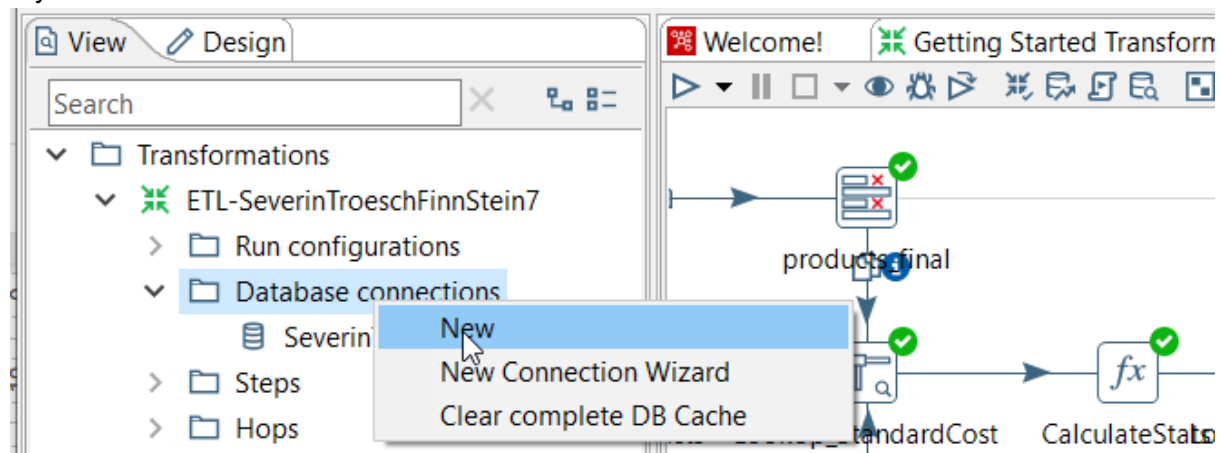
Maximum value:

Help OK Cancel

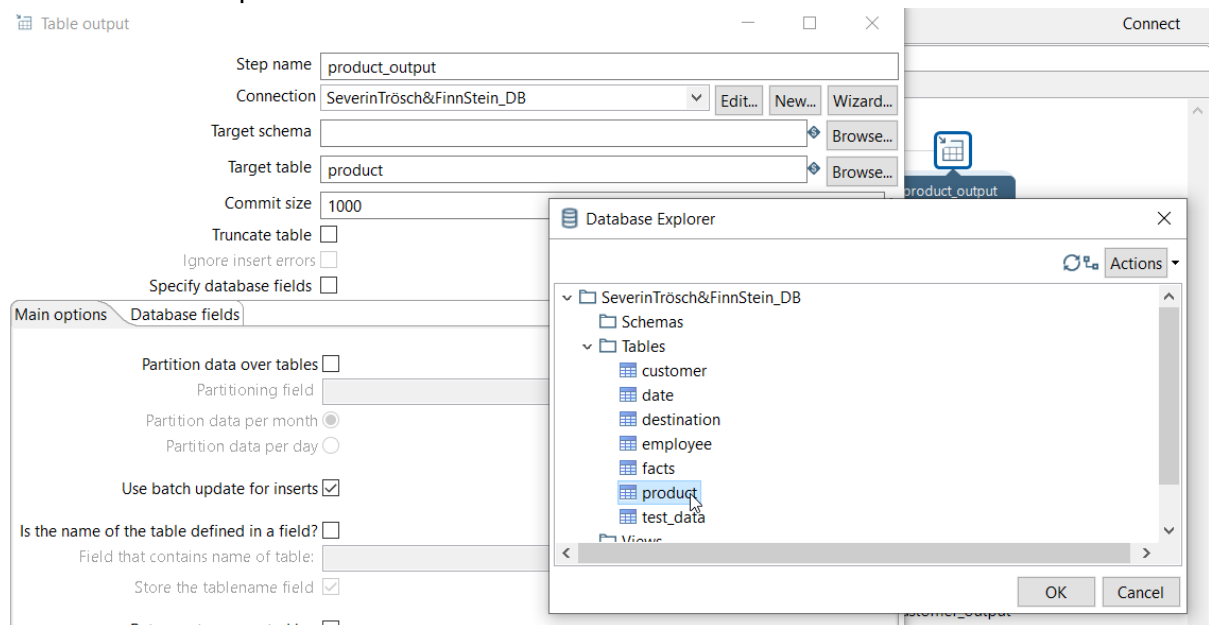
Bei der Tabelle destination ist das Vorgehen analog zur Tabelle date.

A.6 Export der Tabellen in die zuvor erstellte MySQL-Datenbank

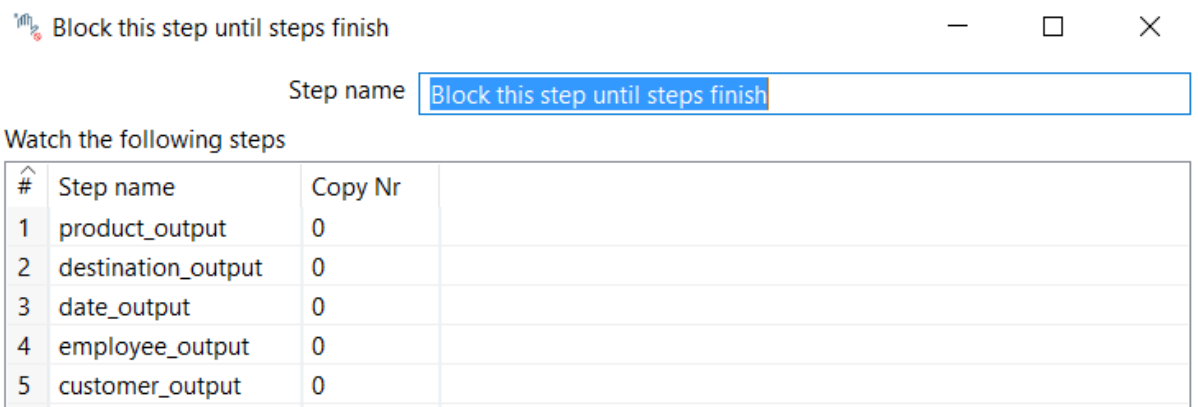
Über "View" => "Database connections" richten wir eine Verbindung zur zuvor erstellten MySQL Datenbank ein.



Anschließend exportieren wir alle Tabellen über den Step “Output” => “Table output” in die entsprechende Tabelle der zuvor erstellten Datenbank. Aufgrund der Datenbankverknüpfung lässt sich die entsprechende Zieltabelle direkt auswählen.



Die Tabelle “facts” kann aufgrund der Fremdschlüssel Constraints nur hochgeladen werden, nachdem alle anderen Tabellen bereits hochgeladen sind. Daher richten wir über den Step “Flow” => “Block this step until steps finish” einen Block ein, der festlegt, dass zuerst alle anderen “load table” steps beendet sein müssen.



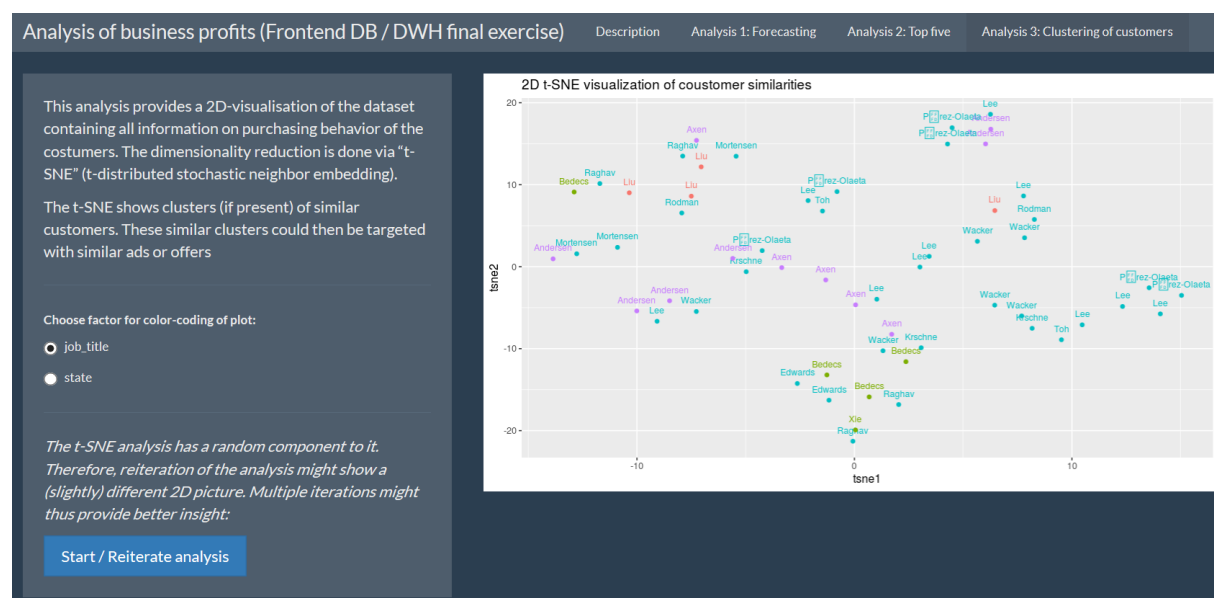
B) Frontend (Analysen & Präsentation)

Erstellen Shiny Web Applikation

Für das interaktive, webbasierte Frontend verwenden wir eine “Shiny App” - ein von RStudio (IDE für die Statistiksoftware R (R version 3.6.1, RStudio Version 1.2.5019)) angebotener Service zur online-Publikation von in R geschriebenen Anwendungen (<https://www.shinyapps.io/>).

Die App wird also in RStudio geschrieben. Es sind zwei Skripte notwendig (beide im Zip-File enthalten):

- “ui.R”: Hier wird das User Interface der App definiert. Beispielsweise welche Art der User-Interaktion möglich ist und wo die Texte oder Abbildungen erscheinen. Im folgenden Screenshot der App ist zu sehen, welche UI-Spezifikationen vorgenommen wurden:
 - Textelemente und Wahlmöglichkeiten links
 - Plot der Analyse rechts
 - Titel der App und Faltblätter für die anderen Analysen oben



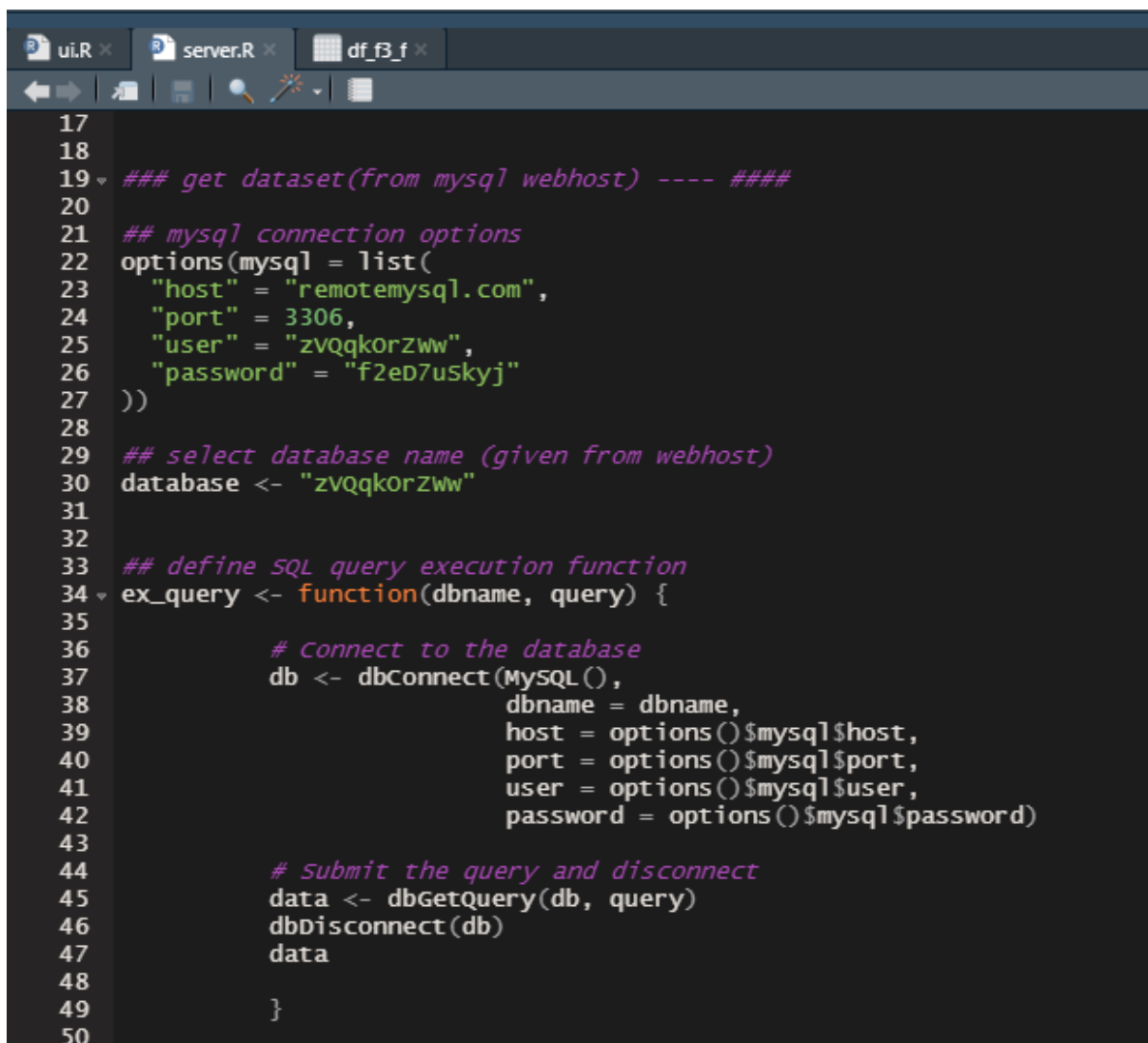
- “server.R”: Hier finden die eigentlichen Analysen statt. Es werden auch die dafür notwendigen R-Bibliotheken geladen und definiert, welche Teile der Analyse auf welchen Teil im User Interface “zeigt” (d.h. nachher in der App sichtbar ist).

Später werden beide Skripte auf einen RStudio Server hochgeladen und sind online zugänglich (s. Unterkapitel “Publikation Shiny Web Applikation”).

Verbindung zu remote MySQL Server

Die Verbindung zum Host unserer MySQL Datenbank findet im Skript "server.R" statt. Es wird mit dem R-Package "RMySQL" gearbeitet. Folgende Schritte sind nötig (s. Abschnitt des R-Codes):

1. Werden die Optionen der MySQL Verbindung definiert (host, port, user und password). Diese sind nötig für die Verbindung mit dem externen Server.
2. Dann wird (in einer von uns definierten R-Funktion) die Verbindung zum Server - gegeben die vorher definierten Optionen - hergestellt ("dbConnect" Funktion).
3. In der gleichen Funktion werden dann auch die SQL-Queries übergeben ("dbGetQuery"). Für die genauen Query-Formulierungen s. nächstes Kapitel.



```
17
18
19 ### get dataset(from mysql webhost) ---- ###
20
21 ## mysql connection options
22 options(mysql = list(
23   "host" = "remotemysql.com",
24   "port" = 3306,
25   "user" = "zvQqk0rZww",
26   "password" = "f2eD7uSkyj"
27 ))
28
29 ## select database name (given from webhost)
30 database <- "zvQqk0rZww"
31
32
33 ## define SQL query execution function
34 ex_query <- function(dbname, query) {
35
36   # Connect to the database
37   db <- dbConnect(MySQL(),
38                   dbname = dbname,
39                   host = options()$mysql$host,
40                   port = options()$mysql$port,
41                   user = options()$mysql$user,
42                   password = options()$mysql$password)
43
44   # Submit the query and disconnect
45   data <- dbGetQuery(db, query)
46   dbDisconnect(db)
47   data
48
49 }
50
```

SQL-Queries in R

Wie oben beschrieben können die eigentlichen, in SQL geschriebenen, Queries dann der eigenen R-Funktion "ex_query" übergeben werden. Der SQL-Code wird dabei mittels paste()-Funktion zusammengeführt. Für jede der drei durchgeführten Analyse wurde eine eigene SQL-Query gemacht (in SQL) und die resultierenden Tabellen dann noch an die spezifischen Bedürfnisse der Analyse angepasst (in R).

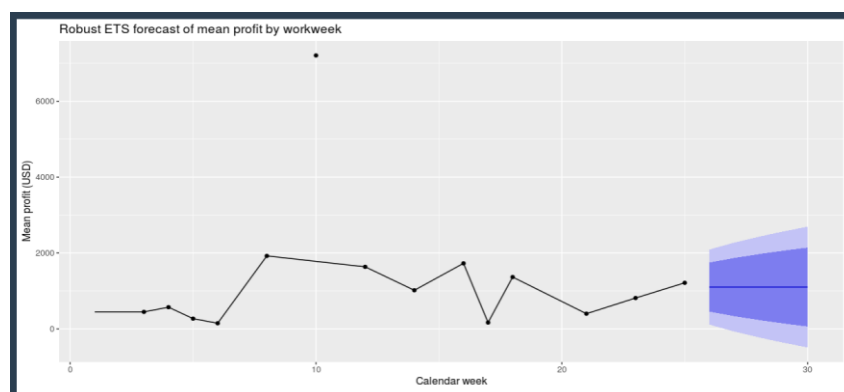
Folgend das Beispiel der SQL-Datenbankabfrage zur dritten Analyse (visuelles Clustering der Verkäufe), die in R geschrieben wurde und aus der in R ein Data Frame ("df_f3") entstand:

```
74
75 # analysis 3:
76 q_f3 = paste("SELECT *",
77             "FROM facts f",
78             "JOIN customer c",
79             "ON f.customer_id = c.id",
80             "JOIN product p",
81             "ON f.product_id = p.id",
82             "JOIN date d",
83             "ON f.date_id = d.id",
84             "ORDER BY f.Gewinn_nach_Discount DESC")
85
86 df_f3 <- ex_query(database,q_f3)
87
```

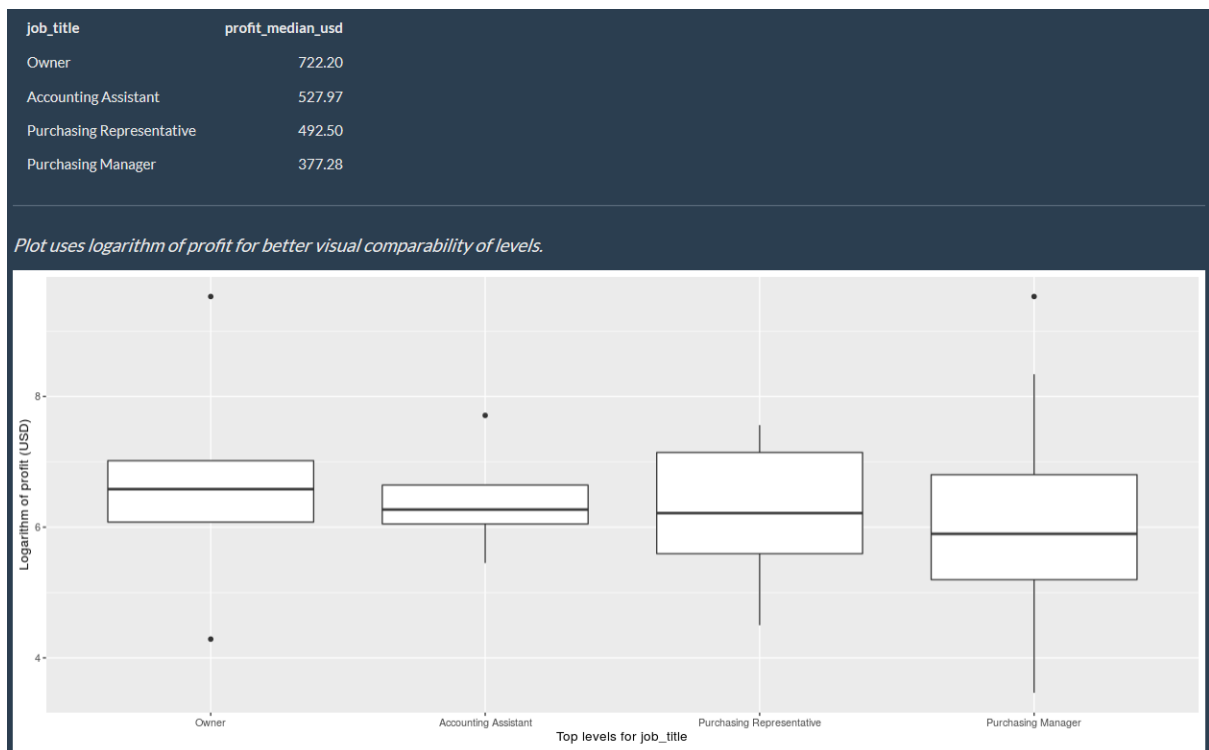
Datenanalyse in R

Basierend auf den Data Frames, die via SQL-Query aus der online MySQL Datenbank gezogen wurden, gingen wir die drei Analysen an. Diese sind in der App dann als eigene Faltblätter (Reiter) wählbar.

- *Analyse 1: Zeitverlauf von des nach Monat aggregierten Gewinns und ein Vorhersage des weiteren Gewinnverlaufs:*
 - Hier wurde mit dem R-Package "forecast" eine robuste Zeitreihe der nach Kalenderwoche aggregierten (gemittelten) Gewinne erstellt. Es wurde die Gewinne nach dem Discount betrachtet. "Robust" bedeutet in diesem Zusammenhang, dass extreme Ausreisser (wie der Gewinn in der KW 10) sowie missing values nicht beachtet werden. Es soll dies eine realistischere Vorhersage ermöglichen (s. blauer Vorhersagebereich in der Graphik unten).
 - Zu beachten ist, dass für die Aggregation nur die tatsächlich passiertten Verkäufe verwendet wurden - und nicht zusätzlich ein "Nullverkauf" an jedem Tag, an dem nichts verkauft wurde. Daher macht die Analyse eher eine Gewinnvorhersage für die Tage, an denen auch verkauft wird statt eine Vorhersage für den erwarteten Gewinn pro Zeit.
 - Folgend die Visualisierung der Analyse aus der App:



- **Analyse 2:** Auflisten (tabellarisch und graphisch) der Top-fünf Gewinnbringer in den Bereichen a) Bundesstaat, b) Kalendermonat und c) Jobbezeichnung.
 - Alle Verkäufe wurde nach den analysierten Bereichen aggregiert und zwar nach Medianwert des Gewinns. Dies zeigt an, beispielsweise bei der Aggregation nach Jobbezeichnung, für welchen Job eines Kunden was für ein Profit bei einem Einkauf typisch ist. Der Median wurde verwendet, da er ein gegen Ausreisser robusteres Lagemass darstellt als der Mittelwert (arithmetisches Mittel).
 - In allen Bereichen wurde dann eine Rangliste nach Median des Gewinns erstellt und (maximal) die besten fünf Levels angezeigt.
 - In der Graphik wurden Boxplots der Top-Levels verwendet, die den Logarithmus des Gewinns und nicht den Gewinn selbst anzeigen. Dies, da die Verteilung der Gewinne stark schief ist (viele Werte nahe bei null und wenige sehr hohe Werte) und die transformierten Werte besser visuell interpretierbar sind (die Log-Transformation ändert die Rangliste nicht).
 - Folgend Tabelle und Graphik zur Rangierung der "job_title" Variable (Jobbezeichnung der Kunden), wie sie in der App angezeigt werden:



- **Analyse 3:** Visuelle Darstellung von allfälligen Kundengruppen, die ähnliches Kaufverhalten zeigen (für kundenspezifische Kampagnen).
 - Für diese Analyse wurde eine "t-SNE" (t-distributed stochastic neighbor embedding) durchgeführt. Dies ist ein Prozess, der es erlaubt, hochdimensionale Daten in 2D darzustellen. Wir stellen hier alle Attribute (Features), die das Kaufverhalten der Kunden betreffen (aus den Tabellen "facts", "products" und "date") in 2D dar.

- [illegible]

Die Publikation erfolgt nach der Verbindung mit der RStudio software mit einem persönlichen Shiny-Konto (hier dies von Severin Trösch), das online erstellt und verwaltet werden kann, direkt aus RStudio.

