Teil 2

Data Wrangling

Rebecca Karwen

Contents

Aufgabe 1

Schreiben Sie eine Query, um den CustomerName und den CustomerZip aller Kunden auszugeben.

Zuerst wurde die Customer Tabelle ausgewählt, der Variable query_1 zugewiesen und danach durch die select Funktion die entsprechenden Spalten ausgewählt.

```
query_1 <- customer %>%
    select(CustomerName, CustomerZip)

query_1
```

```
## # A tibble: 10 x 2
##    CustomerName CustomerZip
##    <chr>              <dbl>
##  1 Dan                55499
##  2 Tina               60137
##  3 Tony               60611
##  4 Pam                35401
##  5 Elly               47374
##  6 Nora               60640
##  7 Miles              60602
##  8 Neil               55403
##  9 Maggie             47401
## 10 Ryan               46202
```

Aufgabe 2

Zeigen Sie die ProductID, den ProductName und den ProductPrice für diejenigen Produkte mit einem ProductPrice von $100 oder höher.

Zunächst wurde durch die filter Funktion nach Reihen mit einem ProductPrice von über hundert gefiltert und dann durch select die entsprechenden Spalten ausgewählt.

```
query_2 <- product %>%
    filter(ProductPrice > 100) %>%
    select(ProductID, ProductName, ProductPrice)
query_2
```

```
## # A tibble: 7 x 3
##   ProductID ProductName   ProductPrice
##   <chr>     <chr>                <dbl>
## 1 1X2       Comfy Harness          150
## 2 1X3       Sunny Charger          125
## 3 4X3       Mega Camera            275
## 4 5X3       Luxo Tent              500
## 5 5X5       Tiny Tent              150
## 6 6X6       Biggy Tent             250
## 7 7X7       Hi-Tec GPS             300
```

Aufgabe 3

Zeigen Sie die ProductID, den ProductName, den ProductPrice und den VendorName für alle Produkte. Sortieren Sie Ihre Ergebnisse anhand der ProductID.

Zunächste wurden alle Produkte durch einen left_join mit der Vendor Tabelle verbunden. Left Join, da wir ja nur den Vendor für die Produkte brauchen, die in der Produkte Tabelle sind, und keine Vendors von denen keine Produkte in der Tabelle sind. Danach wurden durch select die entsprechenden Spalten ausgewählt und durch arrange anhand der ProductID sortiert.

```
query_3 <- product %>%
  left_join(vendor, by = "VendorID", suffix = c("VendorID", "VendorName")) %>%
  select(ProductID, ProductName, ProductPrice, VendorName) %>%
  arrange(ProductID)

query_3
```

```
## # A tibble: 24 x 4
##    ProductID ProductName            ProductPrice VendorName
##    <chr>     <chr>                         <dbl> <chr>
##  1 1X1       Zzz Bag                         100 Pacifica Gear
##  2 1X2       Comfy Harness                   150 Mountain King
##  3 1X3       Sunny Charger                   125 Outdoor Adventures
##  4 1X4       Safe-T Helmet                    40 Pacifica Gear
##  5 2X1       Mmm Stove                        80 Wilderness Limited
##  6 2X2       Easy Boot                        70 Mountain King
##  7 2X3       Reflect-o Jacket                 35 Pacifica Gear
##  8 2X4       Strongster Carribeaner           20 Mountain King
##  9 3X1       Sleepy Pad                       25 Wilderness Limited
## 10 3X2       Bucky Knife                      60 Wilderness Limited
## # ... with 14 more rows
```

Aufgabe 4

Zeigen Sie die ProductID, den ProductName, den ProductPrice, den VendorName und den CategoryName für alle Produkte. Sortieren Sie Ihre Ergebnisse anhand der ProductID.

Zunächst wurden zwei left_joins angewendet, um die Tabellen Vendor und Category mit der Produkttabelle zu verbinden. Left_join, damit alle Produkte noch angezeigt werden - auch diese, die möglicherweise keinen Vendor oder keine Categorys haben. Danach werden die entsprechenden Spalten mit select ausgewählt und durch arrange nach der ProduktID sortiert.

```
query_4 <- product %>%
  left_join(vendor, by = "VendorID", suffix = c("VendorID", "VendorName")) %>%
  left_join(category, by = c("CategoryID" = "CategoryID")) %>%
  select(ProductID, ProductName, ProductPrice, VendorName, CategoryName) %>%
  arrange(ProductID)

query_4
```

```
## # A tibble: 24 x 5
##    ProductID ProductName            ProductPrice VendorName         CategoryName
##    <chr>     <chr>                         <dbl> <chr>              <chr>
##  1 1X1       Zzz Bag                         100 Pacifica Gear      Camping
##  2 1X2       Comfy Harness                   150 Mountain King      Climbing
##  3 1X3       Sunny Charger                   125 Outdoor Adventures Electronics
##  4 1X4       Safe-T Helmet                    40 Pacifica Gear      Cycling
##  5 2X1       Mmm Stove                        80 Wilderness Limited Camping
##  6 2X2       Easy Boot                        70 Mountain King      Footwear
##  7 2X3       Reflect-o Jacket                 35 Pacifica Gear      Cycling
##  8 2X4       Strongster Carribeaner           20 Mountain King      Climbing
##  9 3X1       Sleepy Pad                       25 Wilderness Limited Camping
## 10 3X2       Bucky Knife                      60 Wilderness Limited Camping
## # ... with 14 more rows
```

Aufgabe 5

Zeigen Sie die TID, den CustomerName, und das TDate für Verkaufstransaktionen mit einem Kunden, der ein Produkt mit dem ProductName Dura Boot am 2020-01-06 kauft.

Zunächst wurde die Soldvia Tabelle mit der Product, SalesTransaction und customer Tabelle über einen inner_join verbunden. Inner_join, da nur die Zeilen der Tabellen gebraucht werden, die auch mit allen 4 verknüpfbar sind. Danach wird nach dem ProductName "Dura Boot" und dem TDate "2020-01-06" gefiltert. Zuletzt werden durch select noch die entsprechenden Spalten ausgewählt. Dadurch wird ersichtlich, dass der gesuchte Kunde Dan heißt.

```
query_5 <- sold_via %>%
  inner_join(product, by = "ProductID") %>%
  inner_join(sales_transaction, by = "TID") %>%
  inner_join(customer, by = "CustomerID") %>%
    filter(ProductName == "Dura Boot",
           TDate == "2020-01-06") %>%
  select(TID, CustomerName, TDate)

query_5
```

```
## # A tibble: 1 x 3
##   TID   CustomerName TDate
##   <chr> <chr>        <date>
## 1 T606  Dan          2020-01-06
```

Aufgabe 6

Zeigen Sie für alle Regionen die RegionID, den RegionName und die Anzahl der Stores in der Region. Jede Region darf jedoch nur einmal genannt werden.

Zunächst wird die Region durch einen left_join mit der Store Tabelle verbunden. Ein left_join, da nur die Stores benötigt werden, die auch einer RegionID aus der Region Tabelle zugeordnet werden können, und mögliche Regionen, die keinen Store haben, erhalten bleiben. Danach wird nach Region ID und Region Name gruppiert, damit dann durch die Count Funktion die Anzahl der Stores in der Regionen - also wie oft eine Region vorkommt - errechnet werden kann. Danach wird die entstandene Spalte n noch in anzahl_stores umbenannt, damit gleich der Inhalt zu erkennen ist.

```
query_6 <- region %>%
  left_join(store, by="RegionID") %>%
  select(RegionID, RegionName) %>%
  group_by(RegionID, RegionName) %>%
  count(RegionID) %>%
  rename(anzahl_stores = n)

query_6
```

```
## # A tibble: 4 x 3
## # Groups:   RegionID, RegionName [4]
##   RegionID RegionName  anzahl_stores
##   <chr>    <chr>               <int>
## 1 C        Chicagoland             4
## 2 I        Indiana                 3
## 3 N        North                   3
## 4 T        Tristate                4
```

Aufgabe 7

Zeigen Sie für jede Produktkategorie die CategoryID, den CategoryName und den durchschnittlichen Preis eines Produkts in der Kategorie.

Zunächst wurde wieder ein inner_join gemacht, damit alle CategoryNames mit allen Produkten verknüpft werden, die auch eine Category haben. Danach wird nach CategoryID und CategoryName gruppiert, damit diese im Ergebnis sichtbar sind. Anschließend wird durch die summarize Funktion eine neue Spalte mit dem durchschnittlichen ProductPrice erstellt.

```
query_7 <- category %>%
  inner_join(product, by="CategoryID") %>%
  group_by(CategoryID, CategoryName) %>%
  summarize(mean(ProductPrice))
```

```
## `summarise()` has grouped output by 'CategoryID'. You can override using the `.groups` argument.
```

```
query_7
```

```
## # A tibble: 5 x 3
## # Groups:   CategoryID [5]
##   CategoryID CategoryName `mean(ProductPrice)`
##   <chr>      <chr>                        <dbl>
## 1 CL         Climbing                      66.7
## 2 CP         Camping                      166.
## 3 CY         Cycling                       30
## 4 EL         Electronics                  186.
## 5 FW         Footwear                      59
```

Aufgabe 8

Zeigen Sie die TID für alle Verkaufstransaktionen an, für die die Gesamtzahl der verkauften Artikel (aller Produkte) innerhalb der Transaktion größer als fünf ist.

Hierfür wird nur die solv_via Tabelle benötigt. Zunächst wird nach der TID gruppiert und mit der summarize Funktion die Summe der NoOfItems genommen. Danach wird durch die filter Funktion noch auf Artikel, deren Gesamtzahl an verkauften Produkten höher als 5 ist, beschränkt.

```
query_8 <- sold_via %>%
  group_by(TID) %>%
  summarise(Summe = sum(NoOfItems)) %>%
  filter(Summe > 5)



query_8
```

```
## # A tibble: 10 x 2
##    TID   Summe
##    <chr> <dbl>
##  1 T022      8
##  2 T303      9
##  3 T333      6
##  4 T505      8
##  5 T555      7
##  6 T606     18
##  7 T707      7
##  8 T808      8
##  9 T888      7
## 10 T999     10
```

Aufgabe 9

Zeigen Sie die ProductID und den ProductName des günstigsten Produktes.

Es wird die filter() Funktion benutzt und innerhalb von ihr die min() Funktion, um das Produkt mit dem niedrigsten Preis auszuwählen. Danach werden durch select noch die benötigten Spalten ausgewählt.

```
query_9 <- product %>%
  filter(ProductPrice == min(ProductPrice)) %>%
  select(ProductID, ProductName)



query_9
```

```
## # A tibble: 1 x 2
##   ProductID ProductName
##   <chr>     <chr>
## 1 3X3       Cosy Sock
```

Aufgabe 10

Zeigen Sie die ProductID, den ProductName und den VendorName für Produkte, deren Preis niedriger als der durchschnittliche Preis aller Produkte ist.

Zuerst wird die Product Tabelle durch einen left_join mit der Vendor Tabelle verbunden. Left_join, damit alle Produkte, auch diese welche möglicherweise keinen Vendor haben, in der Auswertung erscheinen. Danach werden alle Regionen, in denen der ProductPrice niedriger ist als der durchschnittliche ProductPrice, gefiltert. Dazu wird innerhalb der filter Funktion die mean() Funktion verwendet, um den durchschnittlichen ProductPrice zu errechnen und ihn danach mit dem ProductPrice des jeweiligen Produktes zu vergleichen. Anschließend werden mithilfe select Funktion die entsprechenden Spalten ausgewählt.

```
query_10 <- product %>%
  left_join(vendor, by = "VendorID") %>%
  filter(ProductPrice < mean(ProductPrice)) %>%
  select(ProductID, ProductName, VendorName)

query_10
```

```
## # A tibble: 17 x 3
##    ProductID ProductName            VendorName
##    <chr>     <chr>                  <chr>
##  1 1X1       Zzz Bag                Pacifica Gear
##  2 1X4       Safe-T Helmet          Pacifica Gear
##  3 2X1       Mmm Stove              Wilderness Limited
##  4 2X2       Easy Boot              Mountain King
##  5 2X3       Reflect-o Jacket       Pacifica Gear
##  6 2X4       Strongster Carribeaner Mountain King
##  7 3X1       Sleepy Pad             Wilderness Limited
##  8 3X2       Bucky Knife            Wilderness Limited
##  9 3X3       Cosy Sock              Mountain King
## 10 3X4       Treado Tire            Outdoor Adventures
## 11 4X1       Slicky Tire            Outdoor Adventures
## 12 4X2       Electra Compass        Mountain King
## 13 4X4       Dura Boot              Pacifica Gear
## 14 5X1       Simple Sandal          Pacifica Gear
## 15 5X2       Action Sandal          Pacifica Gear
## 16 8X8       Power Pedals           Mountain King
## 17 9X9       Trusty Rope            Wilderness Limited
```

Aufgabe 11

Fügen Sie in der Tabelle Vendor die neue Spalte VendorStatus hinzu. Die Vendoren Mountain King, Outdoor Adventures und Wilderness Limited sollen dabei den Status solvent bekommen, während der Vendor Pacifica Gear den Status bankrupt innehaben soll.

Durch die Verwendung der mutate Funktion wurde eine neue Spalte hinzugefügt und durch die Verwendung von case_when wurden die jeweiligen Werte den VendorNames angepasst.

```
query_11 <- vendor %>%
  mutate(VendorStatus = case_when(
    VendorName == "Mountain King" ~ "solvent",
    VendorName == "Outdoor Adventures" ~ "solvent",
    VendorName == "Wilderness Limited" ~ "solvent",
    VendorName == "Pacifica Gear" ~ "bankrupt"))

query_11
```

```
## # A tibble: 4 x 3
##   VendorID VendorName         VendorStatus
##   <chr>    <chr>              <chr>
## 1 MK       Mountain King      solvent
## 2 OA       Outdoor Adventures solvent
## 3 PG       Pacifica Gear      bankrupt
## 4 WL       Wilderness Limited solvent
```

Aufgabe 12

Fügen Sie nun das neue Produkt mit ProductID 10x10, ProductName Flashy Light, ProductPrice 50, VendorID OA und CategoryID CP hinzu.

```
new_row <-

query_12 <- rbind(product,c("10x10","Flashy Light","50", "OA", "CP"))



query_12
```

```
## # A tibble: 25 x 5
##    ProductID ProductName            ProductPrice VendorID CategoryID
##    <chr>     <chr>                  <chr>        <chr>    <chr>
##  1 1X1       Zzz Bag                100          PG       CP
##  2 1X2       Comfy Harness          150          MK       CL
##  3 1X3       Sunny Charger          125          OA       EL
##  4 1X4       Safe-T Helmet          40           PG       CY
##  5 2X1       Mmm Stove              80           WL       CP
##  6 2X2       Easy Boot              70           MK       FW
##  7 2X3       Reflect-o Jacket       35           PG       CY
##  8 2X4       Strongster Carribeaner 20           MK       CL
##  9 3X1       Sleepy Pad             25           WL       CP
## 10 3X2       Bucky Knife            60           WL       CP
## # ... with 15 more rows
```

Aufgabe 13

Fügen Sie nun der Tabelle Product die Spalte product_price_below_or_above_average hinzu, die -1 sein soll, für alle Produkte deren Preis kleiner als der durchschnittliche Preis über alle Produkte ist, 0, wenn der der durchschnittliche Preis dem Produktpreis entspricht und 1 für alle Produkte, deren Preis größer als der durchschnittliche Preis über alle Produkte ist.

Um eine neue Spalte zu der Product Tabelle hinzuzufügen, wurde die Funktion mutate verwendet und ebenfalls die case_when Funktion. Danach wurde, je nachdem ob der Produkt Price unter oder über dem Durchschnitt (mean) liegt, die -1 oder 1 in die jeweilige Spalte eingefügt.

```
query_13 <- product %>%
  mutate(product_price_below_or_above_average = case_when(
    ProductPrice < mean(ProductPrice) ~ -1,
    ProductPrice > mean(ProductPrice) ~ 1))


query_13
```

```
## # A tibble: 24 x 6
##    ProductID ProductName    ProductPrice VendorID CategoryID product_price_belo~
##    <chr>     <chr>                 <dbl> <chr>    <chr>                    <dbl>
##  1 1X1       Zzz Bag                 100 PG       CP                          -1
##  2 1X2       Comfy Harness           150 MK       CL                           1
##  3 1X3       Sunny Charger           125 OA       EL                           1
##  4 1X4       Safe-T Helmet            40 PG       CY                          -1
##  5 2X1       Mmm Stove                80 WL       CP                          -1
##  6 2X2       Easy Boot                70 MK       FW                          -1
##  7 2X3       Reflect-o Jac~           35 PG       CY                          -1
##  8 2X4       Strongster Ca~           20 MK       CL                          -1
##  9 3X1       Sleepy Pad               25 WL       CP                          -1
## 10 3X2       Bucky Knife              60 WL       CP                          -1
## # ... with 14 more rows
```